COMMODORE

AMIGA

500/1000/2000

KNOW HOW

DR. RUPRECHT

KOMMENTIERTES ROM-LISTING

2

RESOURCES UND DEVICES

Dr. Ruprecht

Kommentiertes ROM-LISTING für Commodore Amiga

Band 2

Die Informationen im vorliegenden Buch wurden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht. Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die fotomechanische Wiedergabe und die Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässig.

Commodore Amiga ist eine Produktbezeichnung der Commodore Büromaschinen GmbH. Frankfurt, die wie der Name Commodore Schutzrechte genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis der Schutzrechtsinhaber.

Impressum:

Herausgegeben als Gemeinschaftsproduktion der Firmen:

EDOTRONIK, BIOSYSTEMS, und GIV, München

Redaktionelle Bearbeitung:	Edotronik GmbH · München
Vertrieb:	BIOSYSTEMS SRI GmbH · München
für den Buchhandel:	Mediscript-Verlag · München
Druck:	Treiber-Offset · München
Umschlaggestaltung:	Grafik-Design W. Schmid · München

ISBN 3-88320-169-3

© 1987 by Mediscript-Verlag · 8000 München 45 · Heidemannstr. 29

Inhaltsverzeichnis

Devices und Resources des Amiga

1 Vorbemerkung 7

2 Allgemeines über Devices und Resources 8

3 Beschreibung der einzelnen Devices 12

3.1 Das Keyboard.device 13
3.2 Das Gameport.device 16
3.3 Das Timer.device 21
3.4 Das Input.device 24
3.5 Das Console.device 27
3.6 Das Trackdisk.device 29

4 Beschreibung der einzelnen Resources 35

4.1 Die Keymap.resource 35
4.2 Die cia.resource 37
4.3 Die Potgo.resource 39
4.4 Die Disk.resource 40
4.5 Die Misc.resource 42

Abkürzungen 43

Resident-Header und gemeinsame Routinen		44
Keyboard.device	Datenbereich	54
	Routinen	57
Gameport.device	Datenbereich	68
	Routinen	72
Input.device	Datenbereich	90
	Routinen	98
Console.device	Datenbereich	116
Console.unit	Datenbereich	119
Console.device	Routinen	123
Keymap USA		197
Keymap D		205
Timer.device	Datenbereich	213
	Routinen	217
Trackdisk.device	Datenbereich	237
Trackdisk.unit	Datenbereich	240
Trackdisk.device	Routinen	245
Ciaa.resource	Datenbereich	305
Ciab.resource	Datenbereich	308
Cia.resource	Routinen	311
Potgo.resource	Datenbereich	317
	Routinen	318
Disk.resource	Datenbereich	322
	Routinen	325
Misc.resource	Datenbereich	333
	Routinen	334
Register		337

Devices und Resources des Amiga

1 Vorbemerkung

Bei der sprachlichen Darstellung von Sachverhalten und Vorgängen, die den Computer betreffen, muß sehr oft entschieden werden, ob ein englischer Ausdruck ins Deutsche übersetzt werden soll oder nicht. In diesem Buch wird dabei der Grundsatz beachtet, daß Übertreibungen in jeder Richtung vermieden werden. Englische Fachausdrücke, die in Programmiererkreisen allgemein bekannt sind, werden also nicht übersetzt, weil in der Regel sonst die Verständlichkeit leidet (z.B. wird der Ausdruck "Carry-Flag" weder schöner noch verständlicher, wenn man stattdessen "Übertragskennzeichen" schreibt); andererseits spricht aber auch nichts dafür, z.B. den Anwender zum "Enduser" zu verunstalten.

Schlüsselbegriffe des Amiga-Betriebssystems werden grundsätzlich in der Form der Original-Dokumentation verwendet. Soweit es notwendig oder sinnvoll erscheint, wird im Beschreibungstext eine kurze Erläuterung dazu gegeben.

Für die Darstellung der Listings gilt folgendes: Die in der ersten Spalte stehende vierstellige Hex-Zahl ist die Adresse des gelisteten Befehls ohne die beiden höchsten Stellen. Wenn auf diese Adresse irgendwo zugegriffen wird, so steht in der zweiten Spalte die vollständige Adresse; diese Einträge sind also als Labels zu verstehen.
In den Befehlsoperanden sind alle Zahlenangaben im Hexagesimalsystem gemacht, deshalb ist das an sich obligatorische $-Zeichen immer weggelassen. Die Lesbarkeit wird dadurch deutlich verbessert.
In den Kommentaren sind alle Zahlenangaben ohne $-Zeichen im Dezimalsystem, solche mit $-Prefix im Hexagesimalsystem gemacht. Eine Ausnahme bilden vollständige Adressen (= Labels), die kein $-Zeichen tragen.

Bei der Darstellung von Datenbereichen im RAM sind in der ersten Spalte die Offsets zum Bereichsanfang bzw. zur LibraryBase als vierstellige Hex-Zahl angegeben. Für Strukturen innerhalb der Datenbereiche, z.B. IORequests oder MessagePorts, sind in der zweiten Spalte die Offsets innerhalb der Struktur zweistellig eingetragen. Die angegebenen Datenwerte sind in den meisten Fällen beispielhaft zu

verstehen; bei Zeigern auf Adressen innerhalb desselben Datenbereichs ist der Offset mit einem Stern davor angegeben.

Alle Adressenangaben dienen nur zur Referenz! Der Aufruf einer Systemroutine über ihre absolute Adresse ist ein schwerer Verstoß gegen die "Philosophie" des Amiga-Systems und kann zu nicht vorhersehbaren Problemen führen.

2 Allgemeines über Devices und Resources

"Device" ist das englische Wort für Gerät oder Vorrichtung, "Resource" bedeutet Hilfsquelle. Im Amiga-Betriebssystem sind die Devices Sammlungen von Routinen und Daten, die für die Kommunikation des Systems mit der Außenwelt benötigt werden. Die Resources sind ähnliche, untergeordnete Sammlungen von Routinen und Daten, die von den Devices und auch von anderen Teilen des Betriebssystems verwendet werden. Das Kickstart-ROM enthält folgende Devices und Resources:

```
keyboard.device          keymap.resource
gameport.device          cia.resource
input.device             potgo.resource
console.device           disk.resource
timer.device             misc.resource
trackdisk.device
audio.device
```

Mit Ausnahme des audio.device sind sie in diesem Buch vollständig gelistet und erläutert.
Einige weitere Devices befinden sich auf der System-Disk und werden bei Bedarf in den Arbeitsspeicher geladen:

```
parallel.device          narrator.device
serial.device            clipboard.device
printer.device
```

Devices und Resources sind spezielle Arten von Libraries und haben prinzipiell denselben Aufbau wie diese. Sie bestehen aus einem unveränderlichen "Kern", der bei

den Kickstart-Devices und -Resources im ROM steht, und einem immer im RAM stehenden Arbeitsbereich, der bei der Initialisierung eingerichtet wird. Dieser Teil hat folgende Form:

```
        jmp ...                 Sprungliste
        jmp ...
         :
         :
        jmp ...
DC.L    ln_Succ   \ lib_Node    Knoten
DC.L    ln_Pred   |
DC.B    ln_Type   |             Typ = 3 (nt_Device)
DC.B    ln_Pri    |             Priorität
DC.L    ln_Name   /             Zeiger auf Device-Namen
DC.W    lib_Flags               Flags für Prüfsumme u.a.
DC.W    lib_NegSize             Länge des Teils vor dem Knoten (Bytes)
DC.W    lib_PosSize             Länge des Teils ab dem Knoten (Bytes)
DC.W    lib_Version             Versionsnummer
DC.W    lib_Revision            Revisionsnummer
DC.L    lib_IdString            Zeiger auf Identifikationstext
DC.L    lib_Sum                 Prüfsumme der Sprungliste
DC.W    lib_OpenCnt             Zähler für Library-Benutzer
        :                       Beginn des Datenbereichs
        :
        :
```

Alle Devices sind mittels ihrer Knoten in der Device-Liste, deren Header im Exec-Datenbereich (Offset $15E) steht, miteinander verknüpft. Für die Resources gibt es eine eigene Resource-Liste, ihr Header hat den Offset $150.

Die Sprungliste vor dem Knoten enthält bei allen Devices folgende Einträge:

```
Offset   Aufruf Funktion
-----------------------------------------
-$24     jmp AbortIO     Ein-/Ausgabe abbrechen
-$1E     jmp BeginIO     Ein-/Ausgabe starten
-$18     jmp Extfunct    Reserviert
-$12     jmp Expunge     Device entfernen
-$ØC     jmp Close       Device schließen
-$Ø6     jmp Open        Device öffnen
```

Davor können noch spezielle weitere, vom jeweiligen Device abhängige Aufrufe stehen. Die Resources haben keine einheitliche Sprungliste. Auch der Datenbereich hat bei jedem Device und jeder Resource einen spezifischen Aufbau. Diese Details werden bei der Beschreibung der einzelnen Devices und Resources eingehend behandelt.

Ein Device wird zur Arbeit veranlaßt durch Übersendung einer Message namens IORequest (Ein-/Ausgabe-Auftrag) von folgender Gestalt:

```
DC.L    ln_Succ  \      Knoten
DC.L    ln_Pred   |
DC.B    ln_Type   |     Typ = 5 (nt_Message) oder 7 (nt_ReplyMsg)
DC.B    ln_Pri    |     Priorität (in der Regel Ø)
DC.L    ln_Name  /      Zeiger auf Namen oder Ø
DC.L    mn_ReplyPort    Zeiger auf Antwort-Port
DC.W    mn_Length       Länge der nachfolgenden Message
DC.L    io_Device       Zeiger auf das angesprochene Device
DC.L    io_Unit         Zeiger auf Unit bzw. MsgPort des Device
DC.W    io_Command      Das vom Device auszuführende Kommando
DC.B    io_Flags        Flagbits für verschiedene Zwecke
DC.B    io_Error        Fehlerkode oder Ø
```

Hier schließt sich in der Regel eine Erweiterung (Extension) an, die deviceabhängig ist.

Die Standard-Erweiterung sieht so aus:

```
DC.L    io_Actual       Ergebnis des Auftrags
DC.L    io_Length       Zahl der zu übertragenden Bytes
DC.L    io_Data         Zeiger auf Datenbereich
DC.L    io_Offset       Byte-Offset (z.B. im File)
```

Diese Struktur wird als IOStdRequest (IORequest mit Standard-Erweiterung) bezeichnet.

Wenn ein Device verwendet werden soll, muß es zunächst geöffnet werden. Das Öffnen geschieht mittels der Exec-Routine OpenDevice (Vektoroffset -$1BC). Dabei sind folgende Parameter zu übergeben: a0 zeigt auf den Device-Namen, a1 auf einen noch leeren IORequest-Block (d.h. auf einen reservierten Speicherbereich der richtigen Länge), d0 ist die Nummer der "Unit" (Einheit, z.B. Laufwerknummer beim Trackdisk.device; nicht bei allen Devices vorhanden) und d1 sind Flags (normalerweise 0, da nicht benötigt). Die Routine OpenDevice sucht nun in der Device-Liste das Device mit dem angegebenen Namen, trägt im Erfolgsfall den Zeiger auf den Device-Knoten in das io_Device-Feld im IORequest ein, ruft die Open-Routine in der Device-Sprungliste und gibt d0 = 0 sowie io_Error = 0 zurück, wenn alles in Ordnung ist. Andernfalls enthält d0 und io_Error einen Fehlerkode, z.B. -1, wenn das Device nicht gefunden wurde.

Nun kann der Auftraggeber io_Command und die erforderlichen Parameter, die vom Device zur Kommandoausführung benötigt werden, in den IORequest-Block eintragen und über die Device-Sprungliste BeginIO aufrufen, wobei a1 auf den IORequest und a6 auf das Device zeigen muß. Das Device führt dann entweder den Auftrag sofort aus oder stellt ihn in eine Warteliste, wenn es gerade beschäftigt ist. Nach Abschluß der Arbeit wird der IORequest als ReplyMessage (Typ 7) an den ReplyPort zurückgesandt, nachdem vom Device verschiedene Ergebnisdaten eingetragen worden sind. Der Auftraggeben kann am Wert von io_Error erkennen, ob der Auftrag richtig abgewickelt werden konnte oder welcher Fehler aufgetreten ist.

Statt des jeweiligen Device-Aufrufs BeginIO können auch die Exec-Aufrufe DoIO (Auftrag erteilen und Ausführung abwarten) oder SendIO (Auftrag erteilen, aber nicht warten) verwendet werden.

Für alle Devices ist eine einheitliche Standard-Kommandoliste definiert, die durch device-spezifische Kommandos ergänzt ist.

Die Standardkommandos sind

```
0       cmd_Invalid     ungültiges Kommando
1       cmd_Reset       Device in Ausgangszustand setzen
2       cmd_Read        Daten vom Device lesen
3       cmd_Write       Daten ins Device schreiben
4       cmd_Update      Daten vom Device auf Datenträger schreiben
5       cmd_Clear       Datenpuffer löschen
6       cmd_Stop        Ein-/Ausgabe unterbrechen
7       cmd_Start       Ein-/Ausgabe nach Unterbrechung fortsetzen
8       cmd_Flush       Ein-/Ausgabe endgültig abbrechen
```

Nicht alle Standardkommandos sind für alle Devices anwendbar. Wird ein unzulässiges Kommando verwendet, so wird in io_Error der Fehlerkode ioerr_NoCmd = -3 zurückgegeben.

Wird ein Device nicht mehr benötigt, so sollte es mittels des Exec-Aufrufs CloseDevice (Offset -$1C2) wieder geschlossen werden. Erst wenn alle Benutzer, die das Device geöffnet haben, es auch wieder geschlossen haben, kann der von dem Device belegte Speicherbereich wieder freigegeben werden.

3 Beschreibung der einzelnen Devices

Die Library-Kerne von Keymap.resource, Keyboard.device, Gameport.device, Input.device und Console.device bilden im Kickstart-ROM einen geschlossenen Block (FE4FE4 bis FE8884), an dessen Anfang alle Resident-Header des Blocks stehen. In dem darauf folgenden Bereich (FE515C bis FE53E8) stehen Routinen, die von allen oder doch mehreren Devices im Block gemeinsam benützt werden. Dann folgen die Kerne der einzelnen Devices: Keyboard.device (FE53E8), Gameport.device (FE5854), Input.device (FE5F74), Console.device (FE66D8), und schließlich die Keymap.resource für die US-Tastatur (FE83C8).
Die übrigen Devices und Resources sind voneinander und von denen des Blocks unabhängig.

3.1 Das Keyboard.device

Das Keyboard.device ist die Empfangsstelle für die von der Tastatur eingehenden Signale. Im einzelnen spielt sich dabei folgendes ab:
Die Tastatur wird ständig von einem eigenen Prozessor im Tastaturgehäuse überwacht. Wird eine Taste gedrückt oder losgelassen, so sendet dieser Prozessor den zugehörigen Tastenkode bitseriell über die Signalleitung KDAT zum Rechner, wobei über die zweite Signalleitung KCLK gleichzeitig Taktpulse gesendet werden. Mittels der Taktpulse werden die Datenbits in das Schieberegister des CIA- Bausteins 8520A eingelesen. Sobald 8 Bits eingelaufen sind, wird das Datenbyte in das SDR (Serial Data Register) des CIAA kopiert und ein SP-Interrupt (Bit 3 des ICR) generiert. Dieses Interruptsignal wird an die Interrupt-Logik im PAULA- Chip weitergeleitet; sie setzt das Bit 3 im INTREQ-Register ($DFF09C) und sendet ein Interrupt-Signal der Priorität 2 an die CPU, falls Interrupts nicht gerade gesperrt sind.
Die CPU leitet daraufhin die Interruptverarbeitung ein, holt sich den zur Ebene 2 gehörenden Interruptvektor $FC0CA6 von Adresse $68 und ruft damit den einschlägigen Interrupt-Handler. Dieser prüft noch einmal, ob Interrupts freigegeben sind und ob Bit 3 im INTREQR-Register gesetzt ist, und holt dann iv_Data und iv_Code aus dem Exec-Interruptvektor IVPORTS (Offset $78) in die Register a1 und a5. iv_Data zeigt auf den ListHeader der InterruptServer-Kette, die in diesem Fall nur eine einzige Interrupt-Struktur enthält; sie ist ein Teil der ciaa.resource und verweist auf das Keyboard.device und die darin enthaltene InputHandler-Routine (FE571C).
Mittels JMP (a5) wird nun in die Exec-InterruptServer-Schleife gesprungen, aus der nun endlich der InputHandler im Keyboard.device gerufen wird.
Der InputHandler liest den Tastenkode aus dem SDR des CIA-Bausteins, formt ihn geringfügig um zum "RawKeyCode", erzeugt ein Statuswort (Qualifier), das je 1 Bit für die gedrückten Sondertasten (Ctrl, Shift links, Shift rechts, Caps Lock, Alt links, Alt rechts, Amiga links, Amiga rechts) sowie ein Bit für Tasten aus dem Nummernblock enthält, und legt den RawKeyCode (als Wort) und das Statuswort in einem Puffer ab. Dabei wird das Drücken und das Loslassen einer Taste jeweils getrennt registriert.
Der InputHandler führt außerdem im Datenbereich eine Bitmatrix (Offset $106) von 13 Bytes Länge, in der jeder Taste über den RawKeyCode ein Bit zugeordnet ist. Wird eine Taste gedrückt, so wird das zugeordnete Bit gesetzt, wird sie losgelassen, so wird es gelöscht. Diese Matrix kann mit dem Kommando ReadMatrix (io_Command = $A) gelesen, d.h. an eine beliebige Stelle kopiert werden. Die zugehörige Routine beginnt bei FE56AA.

Dies hört sich alles sehr aufwendig an, man möchte glauben, daß entsprechend viel Zeit verbraucht wird. Tatsächlich läuft dieser ganze Vorgang innerhalb von 0.2 ms ab! Um sich eine Vorstellung über die Zeitverhältnisse machen zu können, ist folgende

Überlegung recht nützlich: Denken wir uns alle Vorgänge zeitlich um den Faktor 5000 gedehnt, so dauert das Übertragen eines Tastenkodes von der Tastatur an den Rechner ca. 2.5 Sekunden, die Verarbeitung durch den InputHandler ca. 1 Sekunde, und selbst wenn Sie flotte 300 Anschläge pro Minute auf der Tastatur schaffen, kommt der nächste Tastenkode erst nach mehr als 8 Minuten an. Tastatureingaben beschäftigen den Rechner also nur ganz marginal.

Der Puffer für die RawKeyCodes und Qualifier liegt im Datenbereich des Device (Offset $74) und wird "zyklisch" betrieben: Es gibt unmittelbar vor dem Puffer einen Schreibzeiger (BufferHead) und einen Lesezeiger (BufferTail). Nach der Initialisierung stehen beide auf Null, zeigen also auf den Pufferanfang. Soll ein Eintrag gemacht werden (vgl. FE57F8-FE5822), so wird eine Kopie von BufferHead erzeugt und um die Länge des Eintrags erhöht, so daß sich der neue Wert von BufferHead ergibt. Wird dabei das Pufferende erreicht, so wird dieser neue Wert auf Null zurückgesetzt; wird BufferTail erreicht, so wird BufferHead (alt) um die Länge des Eintrags zurückgesetzt, damit keine früheren Einträge überschrieben werden, und der neue RawKeyCode durch den Überlaufkode $7D ersetzt. Dann wird der neue Eintrag dort gemacht, wohin BufferHead (alt) zeigt, und der neue Wert für BufferHead eingetragen.
Würde man den Puffer ganz voll schreiben, so wäre BufferTail = BufferHead; dies hätte zur Folge, daß man "Puffer leer" und "Puffer voll" nicht unterscheiden könnte. Deshalb wird der letzte, ohne Überschreiben mögliche Eintrag bereits nicht mehr zugelassen.
Soll ein Eintrag aus dem Puffer gelesen werden (vgl. FE5616-FE564A), so wird zunächst geprüft, ob BufferHead = BufferTail ist, dann ist nämlich der Puffer leer und daher nichts zu tun. Andernfalls wird der Eintrag an der Stelle, wohin BufferTail zeigt, gelesen und anschließend BufferTail um die Länge des Eintrags erhöht. Wird dabei das Pufferende erreicht, wird BufferTail auf Null zurückgesetzt.

Das Keyboard.device hat nun nicht nur die Aufgabe, die ankommenden Tastenkodes zu sammeln, sondern es soll sie auf Anforderung mittels IORequest auch wieder herausgeben. Solche Aufträge wickelt die Routine kbd_ReadEvent (FE55C4) ab, zu der, wie die Kommandotabelle FE5506 zeigt, das io_Command 9 führt. Diese Routine liest jeweils einen RawKeyCode und das zugehörige Statuswort aus dem Puffer und erzeugt daraus einen sogenannten InputEvent (Eingabe-Ereignis), den es in dem durch io_Data (Bereichsanfang) und io_Length (Bereichslänge) definierten Speicherbereich ablegt. Dieser Vorgang wird so oft wiederholt, bis entweder keine Kodes mehr im Puffer sind oder im bereitgestellten Speicherbereich kein InputEvent mehr untergebracht werden kann. Dann wird der IORequest als ReplyMsg zurückgeschickt, nachdem noch in io_Actual die Zahl der mit InputEvents belegten Speicherbytes eingetragen wurde.

Ein InputEvent (IE) ist eine Datenstruktur von folgender Gestalt:

```
DC.L    ie_NextEvent        Zeiger auf den (zeitlich) nächsten IE
DC.B    ie_Class            Event-Klasse
DC.B    ie_SubClass         Event-Subklasse (optional)
DC.W    ie_Code             Event-Kode
DC.W    ie_Qualifier        Event-Qualifier
DC.W    ie_X                Mauszählerwert (X)
DC.W    ie_Y                Mauszählerwert (Y)
DC.L    tv_Secs             Zeitstempel ie_TimeStamp (Sekunden)
DC.L    tv_Micro                                     (Mikrosek)
```

Mehrere aufeinanderfolgende InputEvents werden mittels des Zeigers ie_NextEvent miteinander verkettet; im letzten Event einer Kette ist ie_NextEvent = 0. Die kbd_ReadEvent-Routine belegt im übrigen nur ie_Class mit ieclass_RawKey = 1 sowie ie_Code und ie_Qualifier mit den aus dem Puffer gelesenen Werten; ferner werden ie_SubClass, ie_X und ie_Y auf Null gesetzt. Ein Zeitstempel wird vom Keyboard.device nicht eingetragen.

Es gibt im Keyboard.device eine ResetHandler-Liste. Sie soll alle Routinen enthalten, die im Falle eines System-Reset noch ausgeführt werden müssen, bevor die CPU zurückgesetzt wird. Für die Verwaltung der Liste sind die Kommandos AddResetHandler, RemResetHandler und ResetHandlerDone vorgesehen. Der InputHandler startet die Bearbeitung der ResetHandler-Routinen, wenn er von der Tastatur den RawKeyCode $78 empfangen hat (vgl. FE573A). Dies alles wäre eine sehr praktische Einrichtung, denn man könnte vor dem "Absturz" noch beliebige Aufräumungsarbeiten ausführen lassen. Leider ist der ganze Aufwand umsonst: Der RawKeyCode $78 läßt sich auf keine Weise durch Druck einer Taste oder einer Tastenkombinationen erzeugen! Die Reset-Kombination Ctrl-A-A bewirkt einen echten Hard-Reset: Werden diese drei Tasten gleichzeitig gedrückt, so zieht die Tastatur die KCLK-Leitung für eine halbe Sekunde auf Low, und dieser Impuls gelangt über ein Zeitglied direkt an den Reset-Eingang der CPU. Eine softwareseitige Beeinflussung dieses Vorgangs ist nicht möglich. Man kann nur vermuten, daß es sich bei den ResetHandlern um ein Relikt aus einer früheren Entwicklungsstufe des Amiga-Systems handelt.

Eine Bemerkung zur Initialisierung (FE5402): Ihre Hauptaufgabe ist die Einrichtung des Datenbereichs für das Keyboard.device. Außerdem öffnet sie die ciaa.resource, generiert dort den auf den InputHandler zeigenden InterruptVektor, setzt ihn in die

IVPORTS-Liste von Exec und initialisiert den Interrupt. Mit dem ersten Tastatur-Handshake wird schließlich die Kommunikation mit der Tastatur eröffnet.

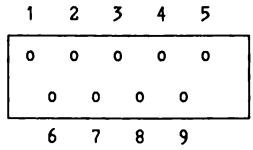
Die verbleibenden Standardroutinen im Keyboard.device (Reset, Clear, Open, Close, Expunge, BeginIO, AbortIO) erklären sich selbst und bedürfen keiner besonderen Beschreibung.

3.2 Das Gameport.device

Das Gameport.device ist die Empfangsstelle für die von den Eingängen 1 (Port 0) und 2 (Port 1) für Maus, Joystick, Trackball, Lightpen u.a. kommenden Signale. Diese Eingänge machen sich nicht, wie die Tastatur, selbst durch einen Interrupt bemerkbar, sie werden vielmehr regelmäßig während des Bildwechselinterrupts, also (im PAL-System) 50 mal in der Sekunde, vom gpd_InputHandler (FE5CC8) abgefragt. Durch den Bildwechsel wird zunächst ein Interrupt der Ebene 5 im PAULA-Chip ausgelöst, was dann zu einem CPU-Interrupt der Priorität 3 führt. Von da bis zum Aufruf des InputHandlers ist der Ablauf ganz analog zum oben beschriebenen Tastatur-Interrupt.

Zunächst ein kurzer Überblick über die Hardware.

Jeder der beiden Eingänge hat - von außen gesehen - folgende Anschlüsse:



Am Stift 7 liegen +5V, Stift 8 liegt an Masse.

Die Stifte 1, 2, 3, 4 führen zum Mauszähler (1: Vertikal-Puls, 2: Horizontal- Puls, 3: Vertikal-Quadratur-Puls, 4: Horizontal-Quadratur-Puls; die Quadratur- Pulse dienen zur Feststellung der Bewegungsrichtung), die dort eingehenden Signale bestimmen den Inhalt der DENISE-Register JOY0DAT (DFF00A) für Port 0 bzw. JOY1DAT (DFF00C) für Port 1, aus denen der InputHandler die Informationen über die Mausbewegung ableitet. Schließt man die Ausgänge eines Joysticks (1: Vor, 2: Zurück, 3: Links, 4: Rechts) an diese Stifte an, wobei die Bewegung des Joysticks den entsprechenden Eingang auf Masse legt, so kann auch der Zustand des Joysticks aus dem Inhalt des einschlägigen Registers JOY0DAT bzw. JOY1DAT ermittelt werden (daher der Name dieser Register).

Die Stifte 5 und 9 sind Potentiometer-Eingänge: Wird einer dieser Stifte über einen Widerstand mit Stift 7 (+5V) verbunden, so kann dieser Widerstand nach folgendem Verfahren gemessen werden: Während des Bildwechsel-Interrupts wird in das PAULA-Register POTGO (DFF034) ein Startkode geschrieben ($0001). Sobald vom nächsten Bildraster 7 oder 8 Horizontallinien geschrieben sind, wird mit der Aufladung eines Kondensators über den Widerstand begonnen und gleichzeitig ein Zähler gestartet, der die von da an geschriebenen Rasterlinien zählt. Er wird gestoppt, sobald die Spannung am Kondensator einen fest eingestellten Referenzwert erreicht hat. Der Zählerstand kann dann während des nächsten Bildwechsels im PAULA-Register POT0DAT (DFF012) für Port 0 bzw. POT1DAT (DFF014) für Port 1 gelesen werden (Bits 0,..,7: Stift 5; Bits 8,..,15: Stift 9). Damit zwischen zwei Bildwechseln kein Zählerüberlauf eintritt, darf der Widerstand nicht größer als 528 kOhm sein.

Die Stifte 5 und 9 können aber auch als Ein- oder Ausgänge für beliebige Zwecke geschaltet werden. Dies geschieht ebenfalls mittels des PAULA-Registers POTGO (DFF034), dessen Bits dafür wie folgt zugeordnet sind:

	Port 1		Port Ø			
	Stift 9	Stift 5	Stift 9	Stift 5		
Bit	15 14	13 12	11 1Ø	9 8	7...1	Ø
	OUT DAT	OUT DAT	OUT DAT	OUT DAT	reserv.	START

Ist ein OUT-Bit = 1, so erscheint der Wert des zugehörigen DAT-Bits am entsprechenden Stift als Ausgang; die Potentiometer-Zählschaltung ist in diesem Fall von dem Stift abgetrennt. Ist dagegen ein OUT-Bit = 0 und das START-Bit = 0, so wird der betreffende Stift als Eingang zu dem zugehörigen DAT-Bit durchgeschaltet. Dieses kann allerdings nicht aus dem POTGO-Register gelesen werden, das ein reines Schreibregister ist; das zugeordnete Leseregister ist POTINP (DFF016), dessen Bits denen von POTGO genau entsprechen. Hieraus ergibt sich die Möglichkeit, diese Stifte für Schalter gegen Masse zu verwenden: Setzt man OUT und DAT für einen Stift auf 1, so erscheint diese 1 am Stift und in dem entsprechenden DAT-Bit von POTINP. Schließt man nun den Schalter an diesem Stift, so wird er auf 0 gezogen und die 0 erscheint im DAT-Bit von POTINP. Die Schalterstellung kann also aus POTINP ausgelesen werden. Dieses Verfahren wird vom gpd_InputHandler benützt, um die rechte (und, soweit vorhanden, die mittlere) Maustaste oder eine Joystick-Taste zu lesen.

Der Stift 6 schließlich ist mit dem CIA-Baustein 8520A verbunden, sein Zustand kann im Register PRA (BFE001) im Bit 6 (Port 1) und Bit 7 (Port 2) abgelesen werden. Er dient außerdem dazu, das Triggersignal eines angeschlossenen Lichtgriffels an den AGNES-Chip zu leiten, wo die aktuelle Position des Bildschirm- Elektronenstrahls festgehalten wird. Damit dies geschieht, muß vorher das LPEN-Bit (Light pen enable, Bit 3 von BPLCON0 = DFF100) gesetzt worden sein.

Nach dieser kurzen Hardware-Beschreibung wenden wir uns nun wieder dem Gameport.device zu. Wie bereits oben festgestellt, wird in jedem Bildwechsel-Interrupt die gpd_InputHandler-Routine aufgerufen. Diese ruft nun zweimal die PortServer-Routine, wobei zuerst der Unit-Zeiger für Port 0, dann der Unit-Zeiger für Port 1 in a0 übergeben wird. Die PortServer-Routine prüft, ob der betreffende Port mittels der Routine SetCType gerade neu konfiguriert wird; ist dies der Fall, so wird die Port-Bedienung sofort abgebrochen. Dann wird ein Zeitzähler (Unit-Offset $70) inkrementiert (falls er schon auf 65535 steht, wird er nicht verändert; dies kann jedoch nur vorkommen, wenn länger als ca. 22 Minuten kein Signal von dem betreffenden Port eingeht). Nun wird geprüft, ob der Controller-Typ für den Port definiert worden ist und einen gültigen Wert hat (1 = Maus, 2 = Relativer Joystick, 3 = Absoluter Joystick); wenn nicht: Abbruch der Routine. Sonst wird die Adresse der einschlägigen Input-Routine aus einer Tabelle (FE5D40) geholt und die Routine aufgerufen.
Die Input-Routine liest zunächst den Zustand der Stifte 5 und 9 aus POTINP (Maustasten 2 und 3, Joystick-Taste 2) und setzt die Bits 29 und 30 von d0 entsprechend. Die Maus-Input-Routine ermittelt sodann die Änderung der Mauszählerwerte seit dem letzten Interrupt und übernimmt das Ergebnis in d0 (Bits 0,..,7: X-Wert; Bits 8,..,15: Y-Wert). Die neuen Mauszählerwerte werden in der Unit (Offset $78) gespeichert. Die Joystick-Input-Routine ermittelt stattdessen die Lage des Joysticks aus den untersten beiden Bits der Mauszählerwerte und übersetzt sie mittels einer Tabelle (FE5F52) in die Werte -1 (Links bzw. Vorn), 0 (Mitte), 1 (Rechts bzw. Hinten) in den Bits 0,..,7 bzw. 8,..,15 von d0. Schließlich lesen beide Routinen noch den Zustand der Maustaste 1 bzw. Joystick-Taste 1 aus dem CIAA-Register PRA (BFE001) und setzen Bit 31 von d0 entsprechend.
Als nächstes prüft der PortHandler, ob die in d0 übergebenen Eingabewerte nicht vielleicht ignoriert werden sollen, weil sie gewisse Schwellenwerte nicht erreichen, die als Trigger-Parameter in der Unit vorgegeben sind. Folgende Möglichkeiten sind vorgesehen: Tastendruck bzw. Taste loslassen wird nur dann registriert, wenn das Downkeys-Flag (Bit 0) bzw. das Upkeys-Flag (Bit 1) in gpt_Keys (Unit-Offset $66) gesetzt ist. Mauszähler-Änderungen werden ignoriert, wenn sie kleiner sind als gpt_XDelta bzw. gpt_YDelta (Unit-Offset $6A bzw. $6C). Falls nach den bisherigen Kriterien keine Triggerung stattfindet, gibt es noch eine Timeout-Schwelle gpt_Timeout, die (falls ungleich 0) doch noch eine Triggerung auslöst, wenn der

Zeitzähler (Unit-Offset $70) den Wert gpt_Timeout erreicht hat. Ist keins der Trigger-Kriterien erfüllt, so beendet der PortHandler seine Arbeit. Andernfalls schreibt er nun die Eingabe in den zyklischen Unit-Puffer (Unit-Offset $26, Schreibzeiger BufferHead $24):
Erstes Byte ist der Tastenkode (LButton = $68, RButton = $69, MButton = $6A, NoButton = -1; bei losgelassener Taste ist Bit 7 gesetzt). Zweites Byte ist ein Statusbyte, in dem für jede gedrückte Maus- oder Joysticktaste ein Bit gesetzt ist; Bit 7 gibt den Zustand des Relativ-Flags wieder, bei Maus und relativem Joystick ist es gesetzt. Die folgenden drei Worte enthalten die X- und Y-Zählerwerte und den aktuellen Zeitzählerstand. Nachdem der Puffer eine Länge von 64 Bytes hat und ein Eintrag 8 Bytes umfaßt, können maximal 8 Einträge gepuffert werden. Werden sie nicht rechtzeitig durch ReadEvent- Requests ausgelesen, so werden die alten Einträge durch neue überschrieben. Nach dem Puffereintrag werden der Zeitzähler und die Variablen, in denen die Zählerwerte bis zur Triggerung aufaddiert werden, auf Null gesetzt. Abschließend wird geprüft, ob am MessagePort ein ReadEvent-Request vorliegt; wenn ja, wird gpd_ReadEvent gerufen.
Sind beide Ports bearbeitet, so wird schließlich vom InputHandler noch das POTGO-Register mit den zugeteilten PotBits belegt; ist das Bit 0 (Startbit) gesetzt, so werden damit die Potentiometerzähler rückgesetzt und der Neustart eingeleitet.

Das Auslesen der Einträge im Unit-Puffer besorgt die Routine gpd_ReadEvent. Sie wird - ebenso wie kbd_ReadEvent - durch einen IORequest mit io_Command = 9 aufgerufen, wobei io_Data und io_Length den Anfang und die Länge des Datenbereichs angeben, in dem die InputEvents, die aus den Puffereinträgen erzeugt werden, abgelegt werden sollen. Ein InputEvent hat die gleiche Gestalt wie oben beschrieben, unterscheidet sich aber durch die Werte der Variablen:
ie_Class = 2 (ieclass_RawMouse), ie_SubClass = 0 oder 1 (Port-Nummer), ie_Code = Maustastenkode (Bit 7 gesetzt, wenn Taste losgelassen), ie_Qualifier = Tastenstatus und Relativ-Flag, ie_X = Mauszählerwert (X), ie_Y = Mauszählerwert (Y), tv_Secs (Bits 16,..,31) = Zeitzählerwert. Die Event-Erzeugung wird solange fortgesetzt, bis der Puffer leer oder der Datenbereich für die Events voll ist. Alle Events sind durch ie_NextEvent verkettet, im letzten Event ist ie_NextEvent = 0. Abschließend wird der IORequest als ReplyMsg zurückgeschickt, nachdem noch in io_Actual die Zahl der mit InputEvents belegten Speicherbytes eingetragen wurde.

Im Gameport.device sind noch folgende weiteren Routinen über IORequests verfügbar:

```
gpd_AskCType     Eingang:  IORequest mit  io_Command = $A
                                          io_Unit -> Unit im Gameport.device
                                          io_Length > Ø (sonst io_Error = -4)
                                          io_Data -> Datenbereich (min. 1 Byte)
                 Ausgang:                 io_Data -> Controller-Typ
                                          io_Actual := 1

gpd_SetCType     Eingang:  IORequest mit  io_Command = $B
                                          io_Unit -> Unit im Gameport.device
                                          io_Length > Ø (sonst io_Error = -4)
                                          io_Data -> CType (1 Byte)
                 Ausgang:                 Unit als Controller konfiguriert
                                          io_Error = 1, falls PotBit-Fehler
                                          io_Actual = 1
                 (CType: 1 = Maus, 2 = RelJoystick, 3 = AbsJoystick)

gpd_AskTrigger   Eingang:  IORequest mit  io_Command = $C
                                          io_Unit -> Unit im Gameport.device
                                          io_Length > 7 (sonst io_Error = -4)
                                          io_Data -> Datenbereich (min. 8 Bytes)
                 Ausgang:                 io_Data -> Triggerparameter (4 Worte)
                                          io_Actual = 8

gpd_SetTrigger   Eingang:  IORequest mit  io_Command = $D
                                          io_Unit -> Unit im Gameport.device
                                          io_Length > 7 (sonst io_Error = -4)
                                          io_Data -> Triggerparameter (4 Worte)
                 Ausgang:                 Triggerparameter in Unit eingetragen
                                          io_Actual = 8
                                          io_Error = Ø (ok) oder -4 (BadLength)
```

Eine Bemerkung zur Initialisierung (FE5854): Ihre Hauptaufgabe ist die Einrichtung des Datenbereichs für das Gameport.device. Außerdem öffnet sie die potgo.resource, fügt die auf den InputHandler zeigende Interrupt-Struktur in die IVVERTB-Liste von Exec ein und initialisiert die Bits 6 und 7 des PRA im CIA-Baustein 8520A.

Die verbleibenden Standardroutinen im Gameport.device (Reset, Clear, Open, Close, Expunge, BeginIO, AbortIO) erklären sich selbst und bedürfen keiner besonderen Beschreibung.

3.3 Das Timer.device

Das Timer.device dient dazu, die vom System oder von Anwenderprogrammen benötigten Zeitinformationen zur Verfügung zu stellen. Es enthält zwei Einheiten (Units): Unit_MicroHz und Unit_VBlank.
Unit_MicroHz verwendet als Zeitgeber den Timer B im CIA-Baustein 8520A, der mit einem Zehntel der CPU-Taktfrequenz 7.15909 MHz betrieben wird; eine Taktperiode (ein Tick) dauert daher 1.40 Mikrosekunden.
Unit_VBlank verwendet als Zeitgeber die Bildwechselfrequenz, die mit der Netzfrequenz 50 Hz übereinstimmt; ein Tick dauert daher hier 20 Millisekunden.

Unit_MicroHz hat eine hohe Auflösung, aber keine sehr gute Langzeit-Genauigkeit. Daher wird diese Einheit vorzugsweise zur Definition kurzer Zeitintervalle verwendet. Unit_VBlank hat eine relativ geringe Auflösung, ist aber über lange Zeiten sehr stabil. Diese Einheit eignet sich also bevorzugt für lange Zeitintervalle.

Das Timer.device wird mittels eines speziellen TimeRequests angesprochen; das ist ein IORequest, dessen Erweiterung eine TimeVal-Struktur ist:

```
                DC.L    tv_Secs         Sekunden
                DC.L    tv_Micro        Mikrosekunden

Folgende Kommandos stehen zur Verfügung:
tr_AddRequest   Eingang: TimeRequest mit io_Command = 9
                                        io_Unit = Ø (MicroHz) oder 1 (VBlank)
                                        tr_Secs   \  Länge des gewünschten
                                        tr_Micro  /  Zeitintervalls

                Ausgang: Der Request wird in die Request-Liste der angegebenen
                         Unit eingeordnet. Wenn das Zeitintervall abgelaufen
                         ist, wird der Request als ReplyMsg zurückgeschickt.

tr_GetSysTime   Eingang: TimeRequest mit io_Command = $A

                Ausgang:                tr_Secs   \  Systemzeit (abhängig vom
                                        tr_Micro  /  letzten SetSysTime)
```

```
tr_SetSysTime   Eingang: TimeRequest mit  io_Command = $B
                                          tr_Secs   \  gewünschter Wert der
                                          tr_Micro  /  Systemzeit
```

Die Routine tr_AddRequest arbeitet folgendermaßen: Ist die Request-Liste leer, so wird der TimeRequest so, wie er ist, in die Liste gesetzt. Sind in der Liste bereits Requests enthalten, so wird der neue Request so eingeordnet, daß die Reihenfolge aller Requests in der Liste der zeitlichen Abfolge ihrer Fälligkeiten entspricht. Die Zeitangaben in den Requests werden dabei jeweils so umgerechnet, daß in jedem Request die Zeit steht, um die er später fällig ist als der vorausgehende Request. Durch dieses Verfahren wird erreicht, daß immer nur der erste Request in der Liste auf seine Fälligkeit geprüft werden muß: Ist seine Zeit abgelaufen, wird er aus der Liste entfernt und zurückgeschickt; damit ist der nächste Request der erste in der Liste.
Die Art der Fälligkeitsüberwachung ist abhängig von der gewählten Unit: In der Request-Liste der Unit_VBlank wird während des Bildwechselinterrupts vom Zeiteintrag des ersten Requests die Zeitspanne zwischen zwei Interrupts (20000 Mikrosekunden) subtrahiert. Ist das Ergebnis negativ oder 0, wird der Request aus der Liste entfernt und in die Reply-Liste gestellt. Dieser Vorgang wird so oft wiederholt, bis die Liste leer oder das Subtraktionsergebnis positiv ist. Abschließend werden alle in der Reply-Liste stehenden Requests zurückgeschickt.
Ein Unit_MicroHz-Interrupt tritt immer dann auf, wenn der abwärts zählende Timer B bei 0 angekommen ist. Wenn ein TimeRequest in die Liste gesetzt worden ist, wird geprüft, ob die (Rest-)Zeit im ersten Request der Liste kleiner oder größer als 91541 Mikrosekunden ist (diese Zeit benötigt der Timer für einen vollen Durchlauf). Ist sie kleiner, so wird der Timer mit dem entsprechenden Anfangswert neu gestartet, sonst mit 65535, und der betreffende Wert wird als Subtrahend gespeichert. Im nächsten Timer-Interrupt wird der Subtrahend von der Zeit im ersten Request abgezogen; ist das Ergebnis Null, so wird der Request in die Reply-Liste gestellt. Im übrigen läuft das Verfahren analog ab wie vorher beschrieben.

Die Routinen tr_GetSysTime und tr_SetSysTime lesen bzw. setzen die Systemuhr. Diese besteht aus einer Zeitbasis im Datenbereich des Timer.device (Offset $C6) und der Tageszeit-Uhr (TOD-Clock, TOD = Time of day) im CIA-Baustein 8520A. Die TOD-Clock besteht aus einem 24-Bit-Zähler, der die Netzfrequenzpulse zählt. SetSysTime setzt die TOD-Clock zurück und belegt die Zeitbasis mit den im IORequest übergebenen Werten tr_Secs und tr_Micro. GetSysTime liest die TODClock und addiert die Zeitbasis. Zu den Mikrosekunden wird eine Korrekturgröße addiert, die dabei jedesmal um 1 erhöht wird; damit wird erreicht, daß zwei Aufrufe von GetSysTime, auch wenn sie unmittelbar aufeinander folgen, mit Sicher-

heit nicht das gleiche Ergebnis haben; die richtige Zeitreihenfolge ist auf diese Weise garantiert.
Die TOD-Clock kann bis 2^24-1 = 16777215 zählen, bevor sie überläuft. Bei einer Taktfrequenz von 50 Hz benötigt sie dazu fast vier Tage. Wenn ein Überlauf eintritt, stimmt die Systemzeit nicht mehr. Um dies zu verhindern, wird in jedem Bildwechsel-Interrupt geprüft, ob das obere Byte der TOD-Clock noch den Wert 0 hat. Ist dies nicht der Fall, wird die TOD-Clock zurückgesetzt, nachdem ihr Inhalt zur Zeitbasis addiert wurde. Ein Überlauf der Zeitbasis selbst erfolgt erst nach 2^32-1 Sekunden, das sind ca. 136 Jahre. AmigaDOS führt in der Zeitbasis auch das Datum mit, indem es jede Eingabe von Datum und Uhrzeit umrechnet auf die Zahl der Sekunden, die seit dem 1. Januar 1978 vergangen sind.

In der Sprungliste des Timer.device sind außer den Standardaufrufen noch die Aufrufe der Routinen AddTime, SubTime und CmpTime enthalten. Dies sind reine Hilfsroutinen, die zum Addieren, Subtrahieren und Vergleichen von TimeVal-Strukturen dienen:

```
Offset  Name            Funktion
------------------------------------------------------------------------
-$02A   AddTime         Addieren von zwei TimeVal-Strukturen
                        Eingang:  a0 -> TimeVal0
                                  a1 -> TimeVal1
                        Ausgang:  a0 -> TimeVal0 = TimeVal0 + TimeVal1

-$030   SubTime         Subtrahieren von zwei TimeVal-Strukturen
                        Eingang:  a0 -> TimeVal0
                                  a1 -> TimeVal1
                        Ausgang:  a0 -> TimeVal0 = TimeVal0 - TimeVal1

-$036   CmpTime         Vergleichen von zwei TimeVal-Strukturen
                        Eingang:  a0 -> TimeVal0
                                  a1 -> TimeVal1
                        Ausgang:  d0 =  -1, falls TimeVal0 < TimeVal1,
                                         0, falls TimeVal0 = TimeVal1,
                                        +1, falls TimeVal0 > TimeVal1.
```

Bei der Initialisierung des Timer.device wird zunächst die zum CIA-Baustein 8520A gehörige ciaa.resource geöffnet. Dann wird der Datenbereich des Device als Library eingerichtet und initialisiert. Anschließend wird der InterruptServer der

Unit_VBlank (FE96DA) in die InterruptServer-Kette gehängt, die dem IVVertB-Interruptvektor (Offset $90 im Exec-Datenbereich) zugeordnet ist, sowie der Interruptvektor zum Timer-B-Interrupt in der ciaa.resource auf den InterruptServer der Unit_MicroHz (FE9726) gerichtet. Schließlich wird die Tageszeit-Uhr (TOD-Clock) im CIA-Baustein rückgesetzt und getestet und mit ihrer Hilfe die Netzfrequenz gemessen (50 oder 60 Hz) und im Exec-Datenbereich (Offset $213) gespeichert. Im Datenbereich des Timer.device wird außer der Netzfrequenz, die gleichzeitig die Bildwechselfrequenz ist, die daraus berechnete Dauer des Bildwechselintervalls in Mikrosekunden (20000 bzw. 16667) eingetragen.

Die verbleibenden Standardroutinen im Timer.device (Open, Close, Expunge, BeginIO und AbortIO) erklären sich selbst und bedürfen keiner besonderen Beschreibung. Wie die Routine Expunge zeigt, läßt sich das Timer.device nicht aus dem System entfernen.

3.4 Das Input.device

Das Input.device hat die Aufgabe, die vom Keyboard.device und Gameport.device gepufferten Eingaben zusammenzuführen und als "Input stream" (Eingabestrom) den Programmen zuzuleiten, für die die Eingaben bestimmt sind. Keyboard.device und Gameport.device sind passive Devices, d.h. sie nehmen nur Daten auf, wenn sie durch ein Interrupt-Signal dazu veranlaßt werden, und sie geben ihre Daten (in Form von InputEvents) nur ab, wenn ein Auftrag (IORequest) dazu vorliegt. Das Input.device kann daher kein passives Device sein, es muß vielmehr laufend IORequests an die beiden anderen Devices senden, den Eingang der Rückmeldungen überwachen und die empfangenen InputEvents weiterleiten. Zentraler Bestandteil des Input.device ist deshalb eine Task (FE62D4), die bei der Initialisierung des Input.device in die Task-Liste des Systems eingefügt wird und dann in der Warteliste schläft, nachdem sie einen MessagePort (für Kommandos) und einen ReplyPort (für Rückmeldungen) mit Signalen versorgt hat. Sobald ein Signal eintrifft, erwacht die Task, sendet Requests an Gameport, Timer und Keyboard und geht dann in eine Schleife, in der sie zunächst, falls vorhanden, den am CommandPort eingegangenen Auftrag bearbeitet. Anschließend geht sie wieder in den Wartestand, bis ein Signal an einem der Ports sie weckt. Sie bearbeitet die eingegangene (Rück-)Meldung, leitet das Ergebnis an die wartenden Empfänger in der InputHandler-Kette weiter, sendet einen neuen IORequest an das betreffende Device, prüft, ob weitere Meldungen anliegen, wiederholt gegebenenfalls den ganzen Vorgang und begibt sich andernfalls wieder zur Ruhe in der Warteliste. Neben den Gameport- und Keyboard-Events, die vom Input.device generiert und an die InputHandler-Kette weitergeleitet werden, erzeugt es in regelmäßigen Abständen von 0.1 Sekunden TimerEvents, indem es entsprechende

TimeRequests an das Timer.device sendet. Die TimerEvents, die durch ie_Class = 6 gekennzeichnet sind und sonst lediglich die aktuelle Systemzeit enthalten, werden ebenfalls an die InputHandler-Kette weitergegeben. Bei den Keyboard-Events kümmert sich das Input.device auch um die Tastenwiederholung: Von jedem wiederholbaren Event wird im Device eine Kopie abgelegt, die, solange die betreffende Taste gedrückt ist, nach Ablauf einer vorgegebenen Anfangsverzögerung (Threshold = Schwelle) in regelmäßigen Zeitabständen (Period) - mit dem aktuellen Zeitstempel versehen - an die InputHandler-Kette weitergeleitet wird.
Die InputHandler-Kette besteht aus Strukturen des Typs Interrupt:

```
DC.L    ln_Succ   \  is_Node     Knoten
DC.L    ln_Pred   |
DC.B    ln_Type   |              Typ = 2 (Interrupt)
DC.B    ln_Pri    |              Priorität bestimmt Platz in der Liste
DC.L    ln_Name   /              Zeiger auf Namen oder Ø
DC.L    is_Data                  Zeiger auf Datenbereich d. InputHandlers
DC.L    is_Code                  Zeiger auf InputHandler-Routine
```

Die InputHandler-Routine ist Bestandteil des Device, der Library oder des Anwenderprogramms, für das die vom Input.device gelieferten Events bestimmt sind. Nach dem Systemstart enthält die InputHandler-Kette nur zwei InputHandler, nämlich den InputHandler von Intuition (ln_Pri = $32) und den des Console.device (ln_Pri = 0). Gibt ein InputHandler d0 = 0 zurück, so werden nachfolgende InputHandler in der Liste nicht mehr gerufen; gibt er dagegen in d0 den Zeiger auf einen Event zurück, so wird dem nächsten InputHandler dieser Event übergeben. Ein InputHandler bekommt also nur diejenigen Events zu sehen, die von den aufgrund ihrer Priorität vor ihm in der Liste stehenden InputHandlern weitergegeben werden. Das Input.device stellt folgende Routinen bereit, mit denen ein InputHandler zur Kette hinzugefügt oder aus der Kette entfernt werden kann:

```
ind_AddHandler   Eingang:  IORequest mit io_Command = 9
                                      io_Data -> InputHandler-Struktur
                 Ausgang:             io_Error = Ø
                 Die InputHandler-Struktur wird gemäß ln_Pri in die Liste
                 eingeordnet.
```

```
ind_RemHandler   Eingang:  IORequest mit  io_Command = $A
                                          io_Data -> InputHandler-Struktur
                 Ausgang:                 io_Error = Ø
                 Die InputHandler-Struktur wird aus der Liste entfernt.
```

An die InputHandler-Kette können auch Events übergeben werden, die nicht durch Keyboard- oder Gameport-Eingaben verursacht worden, sondern von einem beliebigen Programm erzeugt worden sind. Auf diese Weise können beliebige Eingaben simuliert werden. Hierfür steht im Input.device folgende Routine zur Verfügung:

```
ind_WriteEvent   Eingang:  IORequest mit  io_Command = $B
                                          io_Data -> InputEvent
                 Ausgang:                 io_Error = Ø
                 Der InputEvent, auf den io_Data zeigt, wird an die InputHandler-
                 kette übergeben.
```

Bemerkenswert ist die Tatsache, daß die Routinen AddHandler, RemHandler und WriteEvent von der Input-Task im Zuge der Verarbeitung von Requests aufgerufen werden und nicht, wie die nachfolgend aufgeführten Routinen, direkt von der Routine BeginIO. Dadurch wird vermieden, daß das Multitasking ein Durcheinander erzeugt.

Schließlich enthält das Input.device noch Routinen zur Änderung der Tastenwiederholungsparameter und der Gameport-Parameter:

```
ind_SetThresh    Eingang:  TimeRequest mit io_Command = $C
                                           tr_Secs    \  gewünschter Wert der
                                           tr_Micro   /  Anfangsverzögerung

ind_SetPeriod    Eingang:  TimeRequest mit io_Command = $D
                                           tr_Secs    \  gewünschter Wert der
                                           tr_Micro   /  Wiederholungsperiode

ind_SetMPort     Eingang:  IORequest mit  io_Command = $E
                                          io_Length > Ø
                                          io_Data -> MPort (Ø oder 1)
```

```
ind_SetMType     Eingang:  IORequest mit  io_Command = $F
                                          io_Length > 0
                                          io_Data -> MType (1, 2, 3)

ind_SetMTrig     Eingang:  IORequest mit  io_Command = $10
                                          io_Length > 7
                                          io_Data -> MTrig-Werte (4 Worte)
```

Jede der drei letzten Routinen setzt das Gameport-Änderungsflag. Stellt die Input-Task fest, daß dieses Flag gesetzt ist, so schließt sie das GameportDevice, öffnet es wieder und übermittelt ihm die neuen Gameport-Parameter, indem es die Routinen gpd_SetCType und gpd_SetTrigger im Gameport.device ruft. Erst danach werden wieder Meldungen angenommen, die vom Gameport.device kommen.

Die verbleibenden Standardroutinen im Input.device (Reset, Clear, Open, Close, Expunge, BeginIO, AbortIO) erklären sich selbst und bedürfen keiner besonderen Beschreibung. Dasselbe gilt für die Initialisierungs-Routine.

3.5 Das Console.device

Das Console.device verarbeitet die vom Input.device gelieferten Eingabedaten; diese Daten werden jedoch vorher von Intuition "gefiltert", d.h. Intuition besichtigt sie, wählt bestimmte davon (insbesondere die von der Maus erzeugten) zur eigenen Bearbeitung aus und gibt sie unverändert, in modifizierter Form oder gar nicht an das Console.device weiter. Dort werden die Daten vom CDInputHandler (FE6DEC) übernommen. Falls der "Raw Modus" gesetzt ist, erzeugt dieser einen InputEvent-Report, der die unveränderten Daten aus dem InputEvent enthält, andernfalls ruft er die Routine RawKeyConvert (FE71BC). Diese wandelt die RawKeyCodes der normalen Zeichentasten in die entsprechenden ASCII-Kodes, wobei zur Umsetzung unter Berücksichtigung der gleichzeitig gedrückten Sondertasten (Shift, Alt usw.) die Keymap.resource herangezogen wird. Steuertasten, wie die Kursortasten oder die Funktionstasten F1,...,F10, werden in sogenannte Control-Sequenzen umgewandelt; dies sind einigen wenigen Fällen Einzelkodes (z.B. Backspace = 8, Return = 13), meist jedoch Folgen von mehreren Kodes, die von dem CSI-Kode $9B eingeleitet werden (CSI ist die Abkürzung von "control sequence introducer").
Dem Console.device ist für jedes geöffnete Fenster eine Console-Unit zugeordnet, die alle wichtigen Daten über das Fenster sowie einen 256 Bytes langen Puffer für die Tastatureingaben enthält. Alle Eingaben werden der Unit zugeleitet, die zu dem gerade aktiven, d.h. mit dem linken Mausknopf ausgewählten Fenster gehört. Es

obliegt dem Programm, das dieses Fenster geöffnet hat, die Daten im Puffer zu verarbeiten und dafür zu sorgen, daß der Puffer nicht überläuft. Hierfür stellt das Console.device die Routine cd_Read (FE73C0) zur Verfügung, die - auf Anforderung durch einen IORequest - den Puffer ausliest und die Daten an die durch io_Data bestimmte Adresse kopiert.
Der CDInputHandler wird von Intuition ferner über Veränderungen der Fenstergröße informiert und überträgt daraufhin die für die Textausgabe wesentlichen Parameter (Zeilenzahl, Spaltenzahl usw.) in die zugehörige Console-Unit.
Die zweite wesentliche Aufgabe des Console.device ist die Ausgabe von Text auf dem Bildschirm. Sie wird von der Routine cd_Write (FE780C) erledigt. Dabei stehen neben der reinen Textausgabe eine große Zahl von Editorfunktionen zur Verfügung, die von Control-Sequenzen aufgerufen werden können. Die näheren Einzelheiten sind dem Listing zu entnehmen, wobei ein Überblick über die vorhandenen Funktionen aus den Adressentabellen (FE7B12, FE7C66, FE7CD2, FE8012, FE81FC, FE2328) gewonnen werden kann.
Die Routinen CDInputHandler und RawKeyConvert sind über die Device-Sprungliste zu erreichen (Offset -$2A bzw. -$30). Die IO-Kommandos cd_Read und cd_Write wurden bereits erwähnt, die übrigen Standardkommandos bieten keine Besonderheiten. Darüberhinaus sind noch folgende IO-Kommandos möglich:

```
cd_AskKeyMap      Eingang:  IORequest mit  io_Command = 9
                                           io_Unit -> Console-Unit
                                           io_Data -> Datenbereich
                                           io_Length = 32
                  Ausgang:  Die Keymap-Struktur aus der Console-Unit (Offset $42)
                            wird in den Datenbereich kopiert.

cd_SetKeyMap      Eingang:  IORequest mit  io_Command = $A
                                           io_Unit -> Console-Unit
                                           io_Data -> Keymap-Struktur
                                           io_Length = 32
                  Ausgang:  Die Keymap-Struktur, auf die io_Data zeigt, wird in
                            die Console-Unit (Offset $42) kopiert.

cd_AskDefaultKeyMap  E.:  IORequest mit  io_Command = $B
                                           io_Data -> Datenbereich
                                           io_Length = 32
                  Ausgang:  Die Default-Keymap aus dem Console.device (Offset $98)
                            wird in den Datenbereich kopiert.
```

```
cd_SetDefaultKeyMap  E.:  IORequest mit   io_Command = $C
                                          io_Data -> Keymap-Struktur
                                          io_Length = 32
                    Ausgang:  Die Keymap-Struktur, auf die io_Data zeigt, wird in
                              das Console.device (Offset $98) kopiert.
```

Bei der Initialisierung des Console.device wird zunächst der Datenbereich mit Sprungliste und Library-Block eingerichtet. Dann wird die graphics.library geöffnet, aus der einige Routinen für die Bildschirmausgabe benötigt werden. Als nächstes wird die Keymap.resource geöffnet und die erste Keymap-Struktur aus der Liste in den Default-Keymap-Bereich des Console.device (Offset $98) kopiert. Nun wird die vom Console.device geführte Liste der Console-Units initialisiert. Zum Abschluß wird das Input.device geöffnet und dessen Routine ind_AddHandler aufgerufen, die den CDInputHandler in die InputHandler-Liste des Input.device einfügt.

3.6 Das Trackdisk.device

Das Trackdisk.device wickelt den gesamten Datenverkehr mit bis zu vier Floppy-Laufwerken ab. Dabei können sowohl 3.5-Zoll- als auch 5.25-Zoll-Laufwerke bedient werden. In der folgenden Beschreibung des Datenformats beschränken wir uns auf die 3.5-Zoll-Disk.

3.6.1 Das Datenformat auf der Disk

Die Amiga-Diskette wird doppelseitig beschrieben. Sie dreht sich zwischen zwei Schreib-/Leseköpfen, die in radialer Richtung schrittweise bewegt werden können. Jede Kopfstellung entspricht einem "Zylinder", der aus den beiden Magnetspuren auf der Ober- und Unterseite besteht, die sich unter den Köpfen befinden. Eine Diskette enthält 80 Zylinder, numeriert von 0 bis 79. Jede Spur umfaßt 11 Sektoren, numeriert von 0 bis 10, in denen je 512 Bytes gespeichert werden können. Die gesamte Datenkapazität einer Diskette beträgt also 80 * 2 * 11 * 512 = 901120 Bytes. Vor jedem Sektor befindet sich zusätzlich ein Vorspann (Header) mit 32 Bytes Sektor-Informationen.
Die Daten werden nicht in derselben Form auf die Diskette geschrieben, wie sie im Speicher stehen. Aus physikalischen Gründen muß die Magnetisierungsrichtung in einer Spur möglichst oft wechseln, da nur der Wechsel ein Signal im Lesekopf erzeugt. Wenn nun in einer Datenfolge z.B. eine größere Zahl von Nullen aufeinander folgen, würde bei unmittelbarer Aufzeichnung kein Magnetisierungswechsel auftreten und

demzufolge beim Lesen kein Signal entstehen. Damit würde aber jede Information über die Zahl der gelesenen Bits fehlen und es käme ständig zu Lese fehlern. Deshalb werden die Daten vor dem Schreiben in das sogenannte MFM-Format (MFM = Modified Frequency Modulation) umgewandelt nach folgendem Verfahren: Man faßt eine Anzahl von aufeinanderfol- genden Datenbytes zu einer Gruppe zusammen, nimmt alle an geradzahliger Stelle stehenden Bits heraus und fügt sie ohne Änderung der Reihenfolge an die Kette der an ungeradzahliger Stelle stehenden Bits an. Nun setzt man vor das erste Bit und zwischen je zwei weitere Bits der ganzen Kette ein zusätzliches Füllbit, dessen Wert 0 oder 1 so gewählt wird, daß insgesamt möglichst oft zwischen 0 und 1 gewechselt wird. Zum besseren Verständnis betrachten wir ein Beispiel: Die aus den vier Bytes FF,00,01,0B bestehende Gruppe soll in MFM-Format gewandelt werden. Zunächst schreiben wir die ganze Gruppe als Bitmuster:

```
. . . . . . . . . . . . . . . .
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1
```

Die geradzahligen Bits sind durch einen darübergesetzten Punkt gekennzeichnet. Sie werden nun herausgenommen und am Ende der Kette wieder angefügt:

```
                                . . . . . . . . . . . . . . . .
1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1
```

Nun werden die Füllbits eingefügt:

```
0101010001010101010101010001010101010001010101010001000101001
```

Zerlegt man diese Kette wieder in Bytes, so ergibt sich 55,2A,AA,A5,55,2A,A9,29. Wie man sieht, folgen hier - im Gegensatz zum ursprünglichen Format - nie mehr als zwei gleiche Bits aufeinander; der Preis dafür ist eine Verdoppelung der Bytezahl.
Nach dem Lesen der Daten von der Diskette müssen sie aus dem MFM-Format wieder zurückverwandelt werden. Dies ist sehr einfach: Man ersetzt alle Füllbits durch 0, verschiebt die erste Hälfte der Bitreihe um ein Bit nach links und verknüpft dann die beiden Hälften durch die logische Operation OR.
Die Umwandlung in das MFM-Format und zurück wird im Amiga auf Anforderung durch das Trackdisk.device sehr schnell vom Blitter ausgeführt.

Ein Sektor der Amiga-Disk ist in MFM-Kodierung wie folgt gruppiert:

Gruppe	Bytes	MFM-Kode	
1	4	AAAAAAAA	Anfangsmarke
2	4	44894489	2 Sync-Worte
3	8	552AAAA5552AA929	Format, Spur, Sektor, Sektoroffset
4	32	AAAAAAAA....	Sektor-Label (Nutzung optional)
5	8	...	Header-Prüfsumme
6	8	...	Datenblock-Prüfsumme
7	1024	...	Datenblock

In der Gruppe 3 erkennen wir unser obiges Beispiel wieder: Nach der Dekodierung erhält man das Formatbyte $FF (Kennzeichen des Amiga-Formats), Spur 0, Sektor 1, Sektoroffset 11.

Innerhalb einer Spur folgen die Sektoren ohne Zwischenraum aufeinander, die Anfangsmarke des nächsten Sektors schließt sich also unmittelbar an den Datenblock des vorhergehenden an. Nach dem letzten Sektor folgt eine Lücke (Gap), die mit dem MFM-Kode AAAAAA... gefüllt ist. Der Sektoroffset im Header gibt an, wie viele Sektoren noch vor der Lücke liegen; er hat im ersten Sektor nach der Lücke den Wert 11, im letzten vor der Lücke den Wert 1.

3.6.2 Funktion des Trackdisk.device

Für jedes aktive Laufwerk ist dem Trackdisk.device eine Trackdisk-Unit zugeordnet, in der alle für den Zugriff auf das Laufwerk wesentlichen Parameter stehen. Im Datenbereich des Trackdisk.device ist für jede Unit ein Zeiger vorgesehen (Offset $24,...,$30), der für nicht vorhandene Laufwerke den Wert 0 enthält. Der Datenbereich enthält außer einigen für das Device wichtigen Adressen lediglich noch eine ResetHandler-Struktur und einen dazugehörigen IORequest für das Keyboard.device, die aber ziemlich überflüssig sind, weil die ResetHandler nie gerufen werden; hierauf wurde bereits bei der Beschreibung des Keyboard.device eingegangen.
Die Trackdisk-Unit wird vor dem ersten Zugriff auf ein Laufwerk eingerichtet, indem die Exec-Routine OpenDevice mit der entsprechenden Unit-Nummer in D0 aufgerufen wird. Hierbei wird auch ein Spurpuffer für 16388 Bytes MFM-Kode definiert, der mit dem Bitmuster 10101010... gefüllt wird. Wenn sich im Puffer gültige Spurdaten befinden, steht im ersten Wort des Puffers die Spurnummer, sonst -1. Bit 0 im

dritten Byte ist das Änderungsflag: Wurde nach dem Einlesen der Spur etwas geändert, so ist dieses Bit gesetzt, sonst gelöscht. Nach dem vierten Byte bleiben 1660 Bytes frei für die Spurlücke, danach folgen 11*1088 = 11968 Datenbytes im MFM-Format. Beim Schreiben wird mit der Spurlücke am Anfang begonnen, die letzten Daten in der Spur überschreiben dann nach einer vollen Umdrehung der Disk einen gewissen Teil der Leerkodes am Anfang der Lücke. Vor dem Schreiben werden die Sektoroffsets in der Reihenfolge 11, 10,...,1 in die Header eingetragen.
Beim Einlesen einer Spur in den Puffer werden ab einer beliebigen Stelle in der Spur 14716 = 1660+12*1088 Bytes in den Datenbereich des Puffers übernommen. Dadurch ist gesichert, daß alle Sektoren unzertrennt im Puffer stehen. Hierbei wird der Puffer wegen 14716+1660 = 16376 nahezu voll ausgenützt. Die Daten werden nun im allgemeinen noch nicht so im Puffer stehen, daß sie direkt weiterverarbeitet werden können: Erstens werden die Sektoranfänge nicht mit einer Wortgrenze zusammenfallen, sondern um eine unbekannte, zwischen 0 und 15 liegende Zahl von Bits verschoben sein; zweitens wird am Anfang des Datenbereichs kein Sektoranfang stehen; und drittens wird irgendwo im Puffer die Spurlücke sein. Es wird also nun zunächst, von Anfang des Datenbereichs im Puffer beginnend, der erste Sektor-Header gesucht und dabei gleichzeitig die Bitverschiebung festgestellt. Dann wird aus dem Sektoroffset im gefundenen Header der Anfang der Spurlücke berechnet, und schließlich wird noch der Anfang des ersten Sektors nach der Spurlücke gesucht. Mit Hilfe der gewonnenen Ergebnisse wird nun der Inhalt des Datenbereichs so verschoben, daß am Anfang des Datenbereichs ein Sektor-Header beginnt und 11 Sektoren ohne Lücke aufeinanderfolgen. Der erste Sektor wird nun im allgemeinen nicht der Sektor mit der Nummer 0 sein, das ist aber auch nicht unbedingt nötig, da jeder Sektor durch die Header-Information eindeutig identifizierbar ist.
Die beschriebenen Daten-Manipulationen sind relativ aufwendig und würden auch nicht wenig Zeit verschlingen, wenn der Prozessor sie ausführen müßte. Sie werden daher dem Blitter überlassen, der sie wesentlich schneller erledigt, und der Prozessor steht während der Wartezeit anderen Tasks zur Verfügung.

Die Überwachung der IORequest-Eingänge für eine Unit obliegt einer Task, für die bei der Eröffnung in der Unit ein Task-Kontrollblock eingerichtet wird. Alle Units benützen dazu dieselbe Routine (FEAE50), die nicht nur für die Bearbeitung der eingehenden IORequests sorgt, sondern auch die Laufwerke regelmäßig auf einen Diskwechsel prüft. Wird ein Diskwechsel festgestellt, so wird ein Software-Interrupt erzeugt, der zunächst, falls vorhanden, eine vom Anwender einzurichtende Interrupt-Prozedur und dann die in der ChangeInt-Liste der Unit (Offset $18E) verketteten Interrupt-Server aktiviert.

Neben den Standardkommandos, von denen nur Read, Write, Update und Clear zugelassen sind, stellt das Trackdisk.device noch folgende speziellen Kommandos zur Verfügung:

```
td_Motor        Eingang:  IORequest mit  io_Command = $9
                                         io_Length = 0 (aus) oder 1 (ein)
                Ausgang:  Motor aus- bzw. eingeschaltet
                          io_Actual = Motorstatus (0=aus, 1=ein)

td_Seek         Eingang:  IORequest mit  io_Command = $A
                                         io_Offset = Spurnummer * 11 * 512
                Ausgang:  Kopf auf Spur mit eingegebener Nummer,
                          DSKSIDE gesetzt gemäß Spurnummer

td_Format       Eingang:  IORequest mit  io_Command = $B
                                         io_Offset = Anfangsspurnummer * 11 * 512
                                         io_Length = Spurenzahl * 11 * 512
                                         io_Data -> Daten (Bytezahl = io_Length)
                Ausgang:  Daten werden in MFM-Format gewandelt und in die aus
                          io_Offset und io_Length berechneten Spuren geschrieben

td_Remove       Eingang:  IORequest mit  io_Command = $C
                                         io_Data -> Interrupt-Struktur
                Ausgang:  Interrupt-Struktur wird bei Software-Interrupt wegen
                          Diskwechsel aktiviert

td_ChangeNum    Eingang:  IORequest mit  io_Command = $D
                Ausgang:                 io_Actual = Stand des Diskwechselzählers

td_ChangeState  Eingang:  IORequest mit  io_Command = $E
                Ausgang:                 io_Actual =  0: Disk im Laufwerk,
                                                   = -1: Keine Disk im Laufwerk

td_ProtStatus   Eingang:  IORequest mit  io_Command = $F
                Ausgang:                 io_Actual =  0: Kein Schreibschutz,
                                                   = -1: Schreibschutz
```

```
td_RawRead       Eingang:  IORequest mit  io_Command = $1Ø
                                          io_Offset = Spurnummer
                                          io_Data -> Datenbereich
                                          io_Length = Bytezahl n
                 Ausgang:  n Bytes werden von der Spur in den Datenbereich
                           gelesen (Disk-Format). Synchronisation ist möglich.

td_RawWrite      Eingang:  IORequest mit  io_Command = $11
                                          io_Offset = Spurnummer
                                          io_Data -> Daten (Disk-Format)
                                          io_Length = Bytezahl n
                 Ausgang:  n Datenbytes (Disk-Format) werden auf die Spur
                           geschrieben. Synchronisation ist möglich.

td_GetDriveType Eingang:   IORequest mit  io_Command = $12
                 Ausgang:                 io_Actual = 1: 3.5"-Laufwerk
                                                    = 2: 5.25"-Laufwerk

td_GetNumTracks Eingang:   IORequest mit  io_Command = $13
                 Ausgang:                 io_Actual = Zahl der Spuren pro Disk

td_AddChangeInt Eingang:   IORequest mit  io_Command = $14
                                          io_Data -> Interrupt-Struktur
                 Ausgang:  Der IORequest wird an die ChangeInt-Liste der Unit
                           angefügt. Keine Rückmeldung bis td_RemChangeInt!

td_RemChangeInt Eingang:   IORequest mit  io_Command = $15
                 Ausgang:  Der IORequest wird aus der ChangeInt-Liste der Unit
                           entfernt.
```

Die Schreib- und Lesebefehle können durch Setzen von Bit 15 im io_Command modifiziert werden; sie werden dann als "Extended Commands" bezeichnet. In diesem Fall ist der IOStdRequest um zwei Langworte zu erweitern:

```
        DC.L    iotd_Count      Maximum des Diskwechselzählers
        DC.L    iotd_SecLabel   Zeiger auf Daten für den Sektor-Label
```

Das Kommando wird nur dann ausgeführt, wenn der Stand des Diskwechselzählers nicht größer ist als iotd_Count. Ist iotd_SecLabel nicht Null, so werden vor dem Schreiben einer Spur die Daten ab dieser Adresse in Gruppen von jeweils 16 Bytes in MFM-Kode gewandelt und in die Sektor-Header geschrieben; nach dem Lesen wird diese Information wieder dekodiert und ab iotd_SecLabel gespeichert. Auf diese Weise können in die Sektor-Header zusätzliche, vom Anwender zu definierende Informationen geschrieben werden.

4 Beschreibung der einzelnen Resources

4.1 Die Keymap.resource

Die Keymap.resource dient dazu, aus den RawKeyCodes, die von der Tastatur gesendet werden, je nach den Bedürfnissen des Anwenders ASCII-Kodes, Steuerkodes oder Sequenzen aus mehreren Kodes zu erzeugen. Im Kickstart-ROM befindet sich die Original-Keymap für die Tastatur des Amiga 1000 in US-Ausführung. Im Directory "devs" (Devices) der System-Disk befindet sich ein Subdirectory "keymaps", das weitere Keymaps für verschiedene Tastaturen enthält; sie können mittels des AmigaDOS-Befehls "SetMap" geladen und in die ResourceList eingebunden werden. Tatsächlich kann jede Console-Unit ihre eigene Keymap erhalten, so daß dieselbe Taste in verschiedenen Fenstern verschiedene Zeichen erzeugt.

Die immer im RAM stehende Keymap-Struktur besteht aus einem Knoten und den Adressen der acht Tabellen, auf die sich die Keymap-Informationen verteilen. Diese Tabellen teilen sich wieder auf in zwei Gruppen, eine für die "Low Keys", deren RawKeyCodes kleiner als $40 sind, und eine für die "High Keys", das sind alle übrigen. Die Tabellen enthalten folgende Daten:

`km_LowKeyMap` Diese Tabelle enthält für jeden RawKeyCode einen Eintrag von 4 Bytes. Die Bedeutung dieser Bytes wird bestimmt durch den Typ, der in der Tabelle km_LowKeyMapTypes angegeben ist.

km_LowKeyMapTypes Diese Tabelle enthält für jeden RawKeyCode ein Byte, dessen Wert die Bedeutung des Eintrags in der LowKeyMap bestimmt. Folgende Typen sind vorgesehen (T = Taste):

Typ	Byte1	Byte2	Byte3	Byte4
$0	$0	$0	$0	T allein
1	$0	$0	Shift+T	T allein
2	$0	$0	Alt+T	T allein
4	$0	$0	Ctrl+T	T allein
$20	------ Zeiger auf DeadKey-Deskriptor -----			
$40	------ Zeiger auf String-Deskriptor ------			
$80	$0	$0	$0	$0

Der Typ $80 besagt, daß die Taste nicht existiert.
Die übrigen Typen können auch kombiniert werden, so ergibt sich z.B.

3	Shift+Alt+T	Alt+T	Shift+T	T allein

Der Typ 7 ("Vanilla-Key") ist der für Buchstabentasten übliche Typ; er stimmt mit Typ 3 überein, liefert aber zusätzlich für Ctrl+T den ASCII-Kode ohne Bit 6 und Bit 7.

Typ $20 bezeichnet die sog. DeadKeys, die bei Kombination mit Akzenten eine Rolle spielen und aus diesem Grund einer besonderen Behandlung bedürfen.

Typ $40 besagt, daß bei Tastendruck ein String auszugeben ist, auf dessen Deskriptor der Keymap-Eintrag zeigt.

km_LowCapsable Diese Tabelle enthält für jede Taste ein Bit; ist es gesetzt, so wird bei eingeschalteter CapsLock-Taste das Shift-Zeichen geliefert, andernfalls wird die CapsLock-Taste ignoriert.

km_LowRepeatable Diese Tabelle enthält für jede Taste ein Bit; ist es gesetzt, so wirkt bei andauerndem Tastendruck die Wiederholungsfunktion.

km_HiKeyMap, km_HiKeyMapTypes, km_HiCapsable und km_HiRepeatable sind die entsprechenden Tabellen für die "High Keys".

Die Kursortasten, die Funktionstasten F1 bis F10 und noch einige andere Tasten sind StringKeys vom Typ $41 (allein und mit Shift). Die Keymap enthält für diese Tasten einen Zeiger, der auf die zugehörige Stringtabelle verweist. Am Anfang dieser Tabelle stehen zwei Deskriptoren aus je zwei Bytes; das erste gibt die Länge des Strings, das zweite den Offset des Stringanfangs (bezogen auf den Tabellenanfang) an. Die Help-Taste ist von Typ $40, sie kann durch Shift nicht modifiziert werden; daher hat ihre Stringtabelle nur einen Deskriptor.

Die DeadKeys sind Tasten, die sich miteinander zu einem Zeichen kombinieren lassen; hierzu gehören insbesondere die Buchstaben mit Akzent. Wird die Akzenttaste gedrückt, so geschieht zunächst gar nichts. Drückt man anschließend die Taste eines Buchstabens, der mit dem Akzent versehen werden kann, so wird der zugehörige Sonderkode generiert und der Buchstabe mit Akzent erscheint auf dem Bildschirm. Außer mit der auf der deutschen Tastatur vorhandenen Akzenttaste können Akzente und andere Zusätze (Tilde, Trema) durch gleichzeitiges Drücken der Alt-Taste und einer der Tasten f, g, h, i, j erzeugt werden. Sie können kombiniert werden mit a, e, i, o, u, y und n. Für die DeadKeys enthält die Keymap ebenfalls Zeiger, die auf besondere Tabellen verweisen, aus denen der Kombinationskode ermittelt wird.

4.2 Die cia.resource

Die cia.resource dient im wesentlichen zur Verarbeitung der von den beiden CIA-Bausteinen 8520 erzeugten Interrupts: Timer A, Timer B, Alarm, SerialPort, Flag. Die Timer-Interrupts werden dadurch ausgelöst, daß der betreffende Timer beim Abwärtszählen die Null unterschreitet (Underflow = Unterlauf). Der Alarm-Interrupt wird erzeugt, wenn die Tageszeituhr (TODClock) eine einstellbare Zeit, die Alarmzeit, erreicht. Ein SerialPort-Interrupt entsteht, wenn im Input-Modus ein Byte empfangen bzw. im Output-Modus ein Byte gesendet wurde. Der Flag-Interrupt schließlich wird durch eine negative Flanke am FLAG-Eingang des Bausteins bewirkt.

Folgende CIA-Interrupts sind vom System fest zugeordnet:

```
CIAA  Timer B          Timer.device, Unit_MicroHz
      SerialPort       Keyboard.device, InputHandler
      Flag             Parallel.device, Acknowledge (Hardware)

CIAB  Alarm            Graphics.library
      Flag             Trackdisk.device, IndexSync (Hardware)
```

Die cia.resource besteht aus zwei gleichen Library-Strukturen, ciaa.resource und ciab.resource; der zugehörige Library-Kern wird gemeinsam benutzt. Jeder Datenbereich enthält eine Interrupt-Struktur und für jeden der 5 Interrupts einen Interrupt-Vektor. Die Sprungliste enthält 4 Aufrufe:

```
Offset  Name            Funktion
------------------------------------------------------------------------
-$ØØ6   AddICRVector    InterruptVektor in Resource-Datenbereich eintragen
                        Eingang:  a6 -> cia.resource
                                  a1 -> Interrupt-Struktur
                                  dØ = Interrupt-Nummer (Ø,...,4)
                        Ausgang:  InterruptVektor aus Struktur erzeugt,
                                  Interrupt-Enable-Bit im ICR-Register gesetzt
                                  dØ = Ø bei erfolgreichem Abschluß.
                        Falls der InterruptVektor bereits besetzt ist, wird er
                        nicht geändert; in diesem Fall zeigt dØ auf die
                        zugehörige Interrupt-Struktur.

-$ØØC   RemICRVektor    InterruptVektor im Resource-Datenbereich löschen
                        Eingang:  a6 -> cia.resource
                                  dØ = Interrupt-Nummer (Ø,...,4)
                        Ausgang:  InterruptVektor gelöscht,
                                  Interrupt-Enable-Bit im ICR-Register gelöscht
```

```
-$Ø12   AbleICR         Interrupt freigeben/sperren/Enable-Status lesen
                        Eingang:  a6 -> cia.resource
                                  dØ = mask (Bits Ø-4: Int-Bits, Bit 7: SET/CLR)
                                  dØ = Ø, wenn nur gelesen werden soll
                        Ausgang:  Enable-Bits gesetzt/gelöscht falls Bit 7 = 1/Ø
                                  INTREQ gesetzt falls Bit 7 = 1 und Int aktiv
                                  dØ = alter Enable-Status

-$Ø18   SetICR          Interrupt erzeugen/löschen/Interrupt-Status lesen
                        Eingang:  a6 -> cia.resource
                                  dØ = mask (Bits Ø-4: Int-Bits, Bit 7: SET/CLR)
                                  dØ = Ø, wenn nur gelesen werden soll
                        Ausgang:  Status-Bits gesetzt/gelöscht falls Bit 7 = 1/Ø
                                  INTREQ-Bit gesetzt falls Bit 7 = 1
                                  dØ = alter Interrupt-Status
```

Wenn ein CIA-Baustein einen Interrupt generiert, wird der InterruptServer der cia.resource aufgerufen. Dieser odert zunächst das betreffende Interrupt-Flag in die Variable cr_IActive (Offset $29). Dann wird geprüft, ob Interrupts, die in cr_IActive gesetzt sind, auch freigegeben (enabled) sind, indem der Wert von cr_IEnable mit cr_IActive verglichen wird. Falls aktive Interrupts freigegeben sind, werden nacheinander die zugehörigen Interrupt-Routinen über die Interrupt-Vektoren in der cia.resource aufgerufen.

Bei der Initialisierung der cia.resource werden die Datenbereiche für die beiden Libraries erzeugt und initialisiert, die InterruptServer in die zugehörigen Listen eingefügt und die Resources in die Resource-Liste gestellt.

4.3 Die Potgo.resource

Die Potgo.resource vereinfacht den Zugriff auf das Potgo-Register, auf das im Zusammenhang mit dem Gameport.device bereits eingegangen wurde. Es stellt folgende Routinen zur Verfügung:

```
Offset  Name            Funktion
------------------------------------------------------------------------
-$ØØ6   AllocPotBits    Bits im POTGO-Register für Zugriff reservieren
                        Eingang:  a6 -> potgo.resource
                                  dØ = Angeforderte Bits
                        Ausgang:  dØ = Reservierte Bits
                        Die Reservierung ist ganz oder teilweise erfolglos,
                        falls (1) angeforderte Bits bereits reserviert sind,
                        (2) das Startbit bereits reserviert ist, (3) bei
                        Anforderung des Startbits bereits Input-Bits reserviert
                        sind. Über den Erfolg gibt dØ Auskunft.

-$ØØC   FreePotBits     Reservierte Bits im POTGO-Register freigeben
                        Eingang:  a6 -> potgo.resource
                                  dØ = freizugebende Bits

-$Ø12   WritePotgo      Bits im POTGO-Register setzen oder löschen
                        Eingang:  a6 -> potgo.resource
                                  dØ = Datenwort, das geschrieben werden soll
                                  d1 = Maske
                        Ausgang:  Die Bits von dØ, die in d1 gesetzt sind,
                                  werden in das POTGO-Register übertragen und
                                  - mit Startbit = Ø - als aktuelle PotBits
                                  in der Resource (Offset $28) gespeichert.
```

Die Initialisierungsroutine erzeugt lediglich die potgo.resource-Struktur im RAM und fügt sie in die Resource-Liste ein.

4.4 Die Disk.resource

Die Disk.resource unterstützt das Trackdisk.device beim Umgang mit seinen Laufwerken und bedient die zugeordneten InterruptServer (DskBlk, DskSync, Index). Über die Sprungliste sind folgende Aufrufe möglich:

```
Offset  Name            Funktion
------------------------------------------------------------------------------
-$ØØ6   dr_AllocUnit    Laufwerk zuordnen
                        Eingang:  dØ = Laufwerk-Nummer (Ø,...,3)
                        Ausgang:  dØ = -1, wenn erfolgreich
                                  dØ = Ø sonst

-$ØØC   dr_FreeUnit     Laufwerk freigeben
                        Eingang:  dØ = Laufwerk-Nummer (Ø,...,3)

-$Ø12   dr_GetUnit      Laufwerk anfordern
                        Eingang:  a6 -> disk.resource
                                  a1 -> Message in anfordernder Unit
                        Ausgang:  dØ = Ø, wenn Laufwerk beschäftigt, sonst
                                  dØ -> Message in Unit des letzten Benutzers
                        Falls das Laufwerk beschäftigt ist, wird die Message
                        in eine Warteliste gestellt, deren Header in der
                        disk.resource (Offset $4Ø) steht. Sobald das Laufwerk
                        wieder frei ist, erfolgt eine Rückmeldung mit
                        unverändertem Zeiger a1, womit dann erneut GetUnit
                        gerufen werden kann.

-$Ø18   dr_GiveUnit     Laufwerk an nächste Anforderung weitergeben
                        Eingang:  a6 -> disk.resource
                        Ausgang:  Rückmeldung an nächsten Anforderer, falls
                                  vorhanden

-$Ø1E   dr_GetUnitID    Laufwerktyp holen
                        Eingang:  a6 -> disk.resource
                                  dØ = Laufwerknummer (Ø,...,3)
                        Ausgang:  dØ = Laufwerktyp
```

Die Initialisierungsroutine öffnet zunächst die ciab.resource (für den Zugriff auf den Index-Interrupt), erzeugt dann die disk.resource-Library, setzt den DiskBlock- und den DiskSync-InterruptVektor in die InterruptVektor-Tabelle von Exec, erzeugt den Index-InterruptVektor in der ciab.resource, gibt den FLAG-Interrupt und den Disk-DMA frei, initialisiert die Warteliste für die Laufwerkanforderungen in der disk.resource und sucht schließlich alle externen Laufwerke und trägt den Laufwerktyp in die Resource ein.

4.5 Die Misc.resource

Die Misc.resource ist lediglich eine Belegungsliste, in die das zugehörige Device den Zeiger auf den Namen der Task einträgt, die den ParallelPort oder den SerialPort benützen will. In der Liste ist Platz für vier Einträge (MRTypes: mr_SerialPort, mr_SerialBits, mr_ParallelPort, mr_ParallelBits). Ist ein Zeiger eingetragen, so kann sich keine andere Task eintragen. Ist die Arbeit mit dem Port beendet, so wird der Eintrag gelöscht. Für das Eintragen und Löschen gibt es zwei Aufrufe:

```
Offset  Name                Funktion
------------------------------------------------------------------------------
-$006   GetMiscResource     Namenszeiger in Misc.resource eintragen
                            Eingang:  a6 -> misc.resource
                                      a1 = Namenszeiger
                                      d0 = MRType (0,...,3)
                                 Ausgang:  d0 = 0, wenn Zeiger eingetragen
                                      d0 = eingetragener Zeiger, wenn belegt

-$00C   FreeMiscResource    Eintrag in Misc.resource löschen
                            Eingang:  a6 -> misc.resource
                                      d0 = MRType (0,...,3)
```

Die Initialisierungsroutine erzeugt und initialisiert den Datenbereich und fügt die Misc.resource in die Resource-Liste des Systems ein.

Abkürzungen

CD	Console Device
CU	Console Unit
DR	Disk Resource
DRU	Disk Resource Unit
ETD	Extended TrackDisk Command
GPCT	Game Port Controller Type
GPD	Game Port Device
GPT	Game Port Trigger
IE	Input Event
IND	Input Device
KBD	Keyboard Device
KC	Key Code
KM	Key Map
KN	Key Map Node
MR	MiscResource
SGR	Set Graphic Rendition
TD	TrackDisk
TR	Timer
TV	TimeVal

Resident-Header und gemeinsame Routinen

```
4FE4 ;=============================================================
4FE4 ;=                                                           =
4FE4 ;=   Keymap.resource       |                                 =
4FE4 ;=   Keyboard.device       |      Resident-Header             =
4FE4 ;=   Gameport.device       |      und                        =
4FE4 ;=   Input.device          |      gemeinsame Routinen        =
4FE4 ;=   Console.device        |                                 =
4FE4 ;=                                                           =
4FE4 ;=============================================================
4FE4
4FE4 **************  Keymap-Modul  ************************************
4FE4
4FE4 FE4FE4  DC.W    4AFC                ; rt_Matchword
4FE6         DC.L    FE4FE4              ; rt_Matchtag
4FEA         DC.L    FE502E              ; rt_Endskip
4FEE         DC.B    1                   ; rt_Flags
4FEF         DC.B    21                  ; rt_Version
4FF0         DC.B    8                   ; rt_Type
4FF1         DC.B    64                  ; rt_Pri
4FF2         DC.L    FE4FFE              ; rt_Name
4FF6         DC.L    FE500E              ; rt_idString
4FFA         DC.L    FE83C8              ; rt_Init
4FFE
4FFE FE4FFE  DC.B    'keymap.resource',0
500E
500E FE500E  DC.B    'keymap ri 33.103 (1 Aug 1986)',0D,0A,0
502E
502E **************  Keyboard-Modul  **********************************
502E
502E FE502E  DC.W    4AFC                ; rt_Matchword
5030         DC.L    FE502E              ; rt_Matchtag
5034         DC.L    FE507A              ; rt_EndSkip
5038         DC.B    1                   ; rt_Flags
```

```
5039          DC.B    21                  ; rt_Version
503A          DC.B    3                   ; rt_Type (3 = Device)
503B          DC.B    3C                  ; rt_Pri = 60
503C          DC.L    FE5048              ; rt_Name
5040          DC.L    FE5058              ; rt_IdString
5044          DC.L    FE53E8              ; rt_Init
5048
5048 FE5048   DC.B    'keyboard.device',0
5058
5058 FE5058   DC.B    'keyboard ri 33.103 (1 Aug 1986)',0D,0A,0
507A
507A ***************  Gameport-Modul  ***********************************
507A
507A FE507A   DC.W    4AFC                ; rt_Matchword
507C          DC.L    FE507A              ; rt_Matchtag
5080          DC.L    FE50C6              ; rt_EndSkip
5084          DC.B    1                   ; rt_Flags
5085          DC.B    21                  ; rt_Version
5086          DC.B    3                   ; rt_Type (3 = Device)
5087          DC.B    3C                  ; rt_Pri = 60
5088          DC.L    FE5094              ; rt_Name
508C          DC.L    FE50A4              ; rt_IdString
5090          DC.L    FE5854              ; rt_Init
5094
5094 FE5094   DC.B    'gameport.device',0
50A4
50A4 FE50A4   DC.B    'gameport ri 33.103 (1 Aug 1986)',0D,0A,0
50C6
50C6 ***************  Input.Modul  **************************************
50C6
50C6 FE50C6   DC.W    4AFC                ; rt_Matchword
50C8          DC.L    FE50C6              ; rt_Matchtag
50CC          DC.L    FE510E              ; rt_Endskip
50D0          DC.B    1                   ; rt_Flags
50D1          DC.B    21                  ; rt_Version
50D2          DC.B    3                   ; rt_Type (3 = Device)
50D3          DC.B    28                  ; rt_Pri = 40
50D4          DC.L    FE50E0              ; rt_Name
50D8          DC.L    FE50EE              ; rt_IdString
50DC          DC.L    FE5F74              ; rt_Init
```

```
50E0
50E0 FE50E0  DC.B    'input.device',0,0
50EE
50EE FE50EE  DC.B    'input ri 33.103 (1 Aug 1986)',0D,0A,0,0
510E
510E **************  Console-Modul  ***********************************
510E
510E FE510E  DC.W    4AFC                ; rt_Matchword
5110         DC.L    FE510E              ; rt_Matchtag
5114         DC.L    FE66D8              ; rt_Endskip
5118         DC.B    1                   ; rt_Flags
5119         DC.B    21                  ; rt_Version
511A         DC.B    3                   ; rt_Type (3 = Device)
511B         DC.B    14                  ; rt_Pri = 20
511C         DC.L    FE5128              ; rt_Name
5120         DC.L    FE5138              ; rt_IdString
5124         DC.L    FE6B88              ; rt_Init
5128
5128 FE5128  DC.B    'console.device',0,0
5138
5138 FE5138  DC.B    'console ri 33.103 (1 Aug 1986)',0D,0A,0,0,0,0
515C
515C *************** Gemeinsame Routinen für alle Module *****************
515C
515C ;--------------------------------    BeginIO
515C
515C FE515C  movem.l d2/a3,-(a7)         ; Register retten
5160         movea.l a1,a3               ; a3 := a1 -> IORequest
5162         andi.b  #F,1E(a3)           ; io_Flags: Device-Flagbits löschen
5168         clr.b   1F(a3)              ; io_Error löschen
516C         move.w  1C(a3),d2           ; d2 := io_Command
5170         cmp.w   32(a6),d2           ; Kommando im zulässigen Bereich?
5174         blt.s   FE5178              ; Ja: ->
5176         moveq   #0,d2               ; Kommando := 0 (cmd_Invalid)
5178 FE5178  movea.l 2E(a6),a0           ; a0 -> Flag-Tabelle
517C         tst.b   0(a0,d2.w)          ; MsgPort-Nummer = -1?
5180         bmi.s   FE51A0              ; Ja: Warten nicht nötig ->
5182         bsr.s   FE51B4              ; ---> IORequest an MsgPort senden
5184         btst.b  #5,1E(a3)           ; Sofort-Flag gesetzt?
518A         beq.s   FE5198              ; Nein: Muß warten ->
```

```
518A          beq.s   FE5198              ; Nein: Muß warten ->
518C          movea.l 18(a3),aØ           ; aØ := io_Unit
519Ø          btst.b  #Ø,9(aØ)            ; Stop-Flag gesetzt?
5196          beq.s   FE51AØ              ; Nein: ->
5198 FE5198   bclr.b  #Ø,1E(a3)           ; QuickIO-Flag löschen
519E          bra.s   FE51AE              ; ---> Abschluß
51AØ
51AØ FE51AØ   lsl.w   #2,d2               ; d2 := Offset in Adressentabelle
51A2          movea.l 2A(a6),aØ           ; aØ -> Tabelle der Kommandoadressen
51A6          movea.l Ø(aØ,d2.w),aØ       ; aØ := Adresse aus Tabelle
51AA          movea.l a3,a1               ; a1 := a3 -> IORequest
51AC          jsr     (aØ)                ; ---> Kommando ausführen
51AE FE51AE   movem.l (a7)+,d2/a3         ; Register wiederherstellen
51B2          rts
51B4
51B4 ;------ IORequest an Device-MsgPort senden
51B4
51B4 FE51B4   move.l  a3,-(a7)            ; a3 retten
51B6          movea.l a1,a3               ; a3 := a1 -> IORequest
51B8          move.l  a6,-(a7)            ; a6 retten
51BA          movea.l 26(a6),a6           ; a6 := SysBase
51BE          jsr     -78(a6)             ; ---> Disable
51C2          movea.l (a7)+,a6            ; a6 wiederherstellen
51C4          andi.b  #CF,1E(a3)          ; io_Flags: MsgList-Flags löschen
51CA          tst.b   1F(a3)              ; io_Error = Ø?
51CE          bne.s   FE521Ø              ; Nein: Fertig ->
51DØ          move.w  1C(a3),dØ           ; dØ := io_Command
51D4          movea.l 2E(a6),aØ           ; aØ -> Tabelle der MsgPort-Nummern
51D8          move.b  Ø(aØ,dØ.w),dØ       ; dØ := MsgPort-Nummer
51DC          ext.w   dØ                  ; auf Wort erweitern
51DE          mulu    #22,dØ              ; mal 34 (= mp_Size)
51E2          movea.l 18(a3),aØ           ; aØ := io_Unit
51E6          adda.l  dØ,aØ               ; aØ := io_Unit+34*MsgPort-Nummer
51E8          move.l  aØ,-(a7)            ; aØ retten
51EA          movea.l a3,a1               ; a1 := a3 -> IORequest
51EC          move.l  a6,-(a7)            ; a6 retten
51EE          movea.l 26(a6),a6           ; a6 := SysBase
51F2          jsr     -16E(a6)            ; ---> PutMsg
51F6          movea.l (a7)+,a6            ; a6 wiederherstellen
51F8          movea.l (a7)+,aØ            ; aØ wiederherstellen
```

```
51FA          movea.l 14(aØ),aØ           ; aØ = MsgList.lh_Head
51FE          cmpa.l  aØ,a3               ; Steht IORequest am Anfang der Liste?
52ØØ          beq.s   FE52ØA              ; Ja: ->
52Ø2          bset.b  #4,1E(a3)           ; Warte-Flag setzen
52Ø8          bra.s   FE521Ø              ; --->
52ØA
52ØA FE52ØA   bset.b  #5,1E(a3)           ; Sofort-Flag setzen
521Ø FE521Ø   move.l  a6,-(a7)            ; a6 retten
5212          movea.l 26(a6),a6           ; a6 := SysBase
5216          jsr     -7E(a6)             ; ---> Enable
521A          movea.l (a7)+,a6            ; a6 wiederherstellen
521C          movea.l (a7)+,a3            ; a3 wiederherstellen
521E          rts
522Ø
522Ø ;--------------------------------   AbortIO
522Ø
522Ø FE522Ø   moveq   #-2,dØ              ; dØ := ioerr_Aborted
5222          bsr.s   FE5226              ; ---> IORequest beantworten
5224          rts
5226
5226 ;--------------------------------   IORequest beantworten
5226
5226 FE5226   movem.l d2/a3,-(a7)         ; Register retten
522A          move.l  dØ,d2               ; d2 := dØ = Errorkode
522C          movea.l a1,a3               ; a3 := a1 -> IORequest
522E          move.l  a6,-(a7)            ; a6 retten
523Ø          movea.l 26(a6),a6           ; a6 := SysBase
5234          jsr     -78(a6)             ; ---> Disable
5238          movea.l (a7)+,a6            ; a6 wiederherstellen
523A          bset.b  #7,1E(a3)           ; Removed-Flag setzen
524Ø          bne.s   FE5292              ; War schon gesetzt: Fertig ->
5242          move.b  d2,1F(a1)           ; io_Error speichern
5246          btst.b  #5,1E(a3)           ; IORequest am Anfang der Liste?
524C          beq.s   FE5266              ; Nein: ->
524E          movea.l (a3),aØ             ; Macro Remove
5250          movea.l 4(a3),a1            ; :
5254          move.l  aØ,(a1)             ; :
5256          move.l  a1,4(aØ)            ; :
525A          tst.l   (aØ)                ; War das der Tail-Node?
525C          beq.s   FE527C              ; Ja: ->
```

```
525E          bset.b  #5,1E(aØ)          ; Sofort-Flag setzen
5264          bra.s   FE527C             ; --->
5266
5266 FE5266   btst.b  #4,1E(a3)          ; Warte-Flag gesetzt?
526C          beq.s   FE527C             ; Nein: ->
526E          movea.l a3,a1              ; a1 := a3 -> IORequest
527Ø          movea.l (a1),aØ            ; Macro Remove
5272          movea.l 4(a1),a1           ; :
5276          move.l  aØ,(a1)            ; :
5278          move.l  a1,4(aØ)           ; :
527C FE527C   btst.b  #Ø,1E(a3)          ; QuickIO-Flag gesetzt?
5282          bne.s   FE5292             ; Ja: ->
5284          movea.l a3,a1              ; a1 -> IORequest
5286          move.l  a6,-(a7)           ; a6 retten
5288          movea.l 26(a6),a6          ; a6 := SysBase
528C          jsr     -17A(a6)           ; ---> ReplyMsg
529Ø          movea.l (a7)+,a6           ; a6 wiederherstellen
5292 FE5292   move.l  a6,-(a7)           ; a6 retten
5294          movea.l 26(a6),a6          ; a6 := SysBase
5298          jsr     -7E(a6)            ; ---> Enable
529C          movea.l (a7)+,a6           ; a6 wiederherstellen
529E          movem.l (a7)+,d2/a3        ; Register wiederherstellen
52A2          rts
52A4
52A4 ;--------------------------------  Device einrichten
52A4
52A4 FE52A4   jsr     -54(a6)            ; ---> MakeLibrary
52A8          movem.l (a7)+,aØ/a2        ; Register wiederherstellen
52AC          tst.l   dØ                 ; Device eingerichtet?
52AE          beq.s   FE52BA             ; Nein: ->
52BØ          movea.l dØ,a1              ; a1 -> Device Node
52B2          move.l  aØ,22(a1)          ; SegList abspeichern
52B6          jsr     -1BØ(a6)           ; ---> AddDevice
52BA FE52BA   rts
52BC
52BC ;--------------------------------  Device entfernen
52BC
52BC FE52BC   move.l  22(a6),-(a7)       ; SegList auf Stack
52CØ          movea.l a6,a1              ; a1 := a6 -> Device
52C2          move.w  1Ø(a6),dØ          ; dØ := NegSize
```

```
52C6          suba.w  dØ,a1               ; a1 -> Anfang der Sprungliste
52C8          add.w   12(a6),dØ           ; dØ := NegSize + PosSize
52CC          ext.l   dØ                  ; auf Langwort erweitern
52CE          move.l  a6,-(a7)            ; a6 retten
52DØ          movea.l 26(a6),a6           ; a6 := SysBase
52D4          jsr     -D2(a6)             ; ---> FreeMem
52D8          movea.l (a7)+,a6            ; a6 wiederherstellen
52DA          movea.l a6,a1               ; a1 := a6 -> Device
52DC          movea.l (a1),aØ             ; Macro Remove
52DE          movea.l 4(a1),a1            ; :
52E2          move.l  aØ,(a1)             ; :
52E4          move.l  a1,4(aØ)            ; :
52E8          move.l  (a7)+,dØ            ; dØ := SegList
52EA          rts
52EC
52EC ;--------------------------------   Delayed Expunge
52EC
52EC FE52EC   bset.b  #3,E(a6)            ; DelExp-Flag setzen
52F2          moveq   #Ø,dØ               ; dØ := Ø
52F4 FE52F4   rts
52F6
52F6 ;--------------------------------     Close
52F6
52F6 FE52F6   moveq   #Ø,dØ               ; dØ := Ø
52F8          move.l  dØ,18(a1)           ; IOReq.io_Unit := Ø
52FC          move.l  dØ,14(a1)           ; IOReq.io_Device := Ø
53ØØ          subq.w  #1,2Ø(a6)           ; Open-Zähler erniedrigen
5304          bne.s   FE5312              ; Device noch aktiv: ->
5306          btst.b  #3,E(a6)            ; DelExp-Flag gesetzt?
53ØC          beq.s   FE5312              ; Nein: ->
53ØE          jmp     -12(a6)             ; ---> Expunge
5312
5312 FE5312   rts
5314
5314 ;--------------------------------   Ungültiges Kommando
5314
5314 FE5314   moveq   #-3,dØ              ; dØ := ioerr_NoCmd
5316          bsr     FE5226              ; ---> IORequest beantworten
531A          rts
531C
```

```
531C ;--------------------------------  Reset
531C
531C FE531C  bsr     FE532A            ; ---> Stop
5320         bsr     FE5366            ; ---> Flush
5324         bsr     FE533A            ; ---> Start
5328         rts
532A
532A ;--------------------------------  Stop
532A
532A FE532A  movea.l 18(a1),a0         ; a0 -> io_Unit
532E         bset.b  #0,9(a0)          ; Stop-Flag setzen
5334         bsr     FE5226            ; ---> IORequest beantworten
5338         rts
533A
533A ;--------------------------------  Start
533A
533A FE533A  move.l  a1,-(a7)          ; a1 retten
533C         movea.l 18(a1),a0         ; a0 -> io_Unit
5340         bclr.b  #0,9(a0)          ; Stop-Flag löschen
5346         movea.l 14(a0),a1         ; a1 := lh_Head (MsgList)
534A         tst.l   (a1)              ; Liste leer?
534C         beq.s   FE535E            ; Ja: ->
534E         move.w  1C(a1),d0         ; d0 := io_Command
5352         lsl.w   #2,d0             ; mal 4 ergibt Tabellenoffset
5354         movea.l 2A(a6),a0         ; a0 -> Tabelle der Kommandoadressen
5358         movea.l 0(a0,d0.w),a0     ; a0 := Kommandoadresse
535C         jsr     (a0)              ; ---> Kommando ausführen
535E FE535E  movea.l (a7)+,a1          ; a1 wiederherstellen
5360         bsr     FE5226            ; ---> IORequest beantworten
5364         rts
5366
5366 ;--------------------------------  Flush
5366
5366 FE5366  movem.l a1-a2,-(a7)       ; Register retten
536A         movea.l 18(a1),a2         ; a2 -> io_Unit
536E FE536E  movea.l 14(a2),a1         ; a1 -> io_Device
5372         tst.l   (a1)              ; Device geöffnet?
5374         beq.s   FE537C            ; Nein: ->
5376         bsr     FE5220            ; ---> AbortIO
537A         bra.s   FE536E            ; ---> Loop
```

```
537C
537C FE537C  movem.l (a7)+,a1-a2          ; Register wiederherstellen
5380         bsr     FE5226               ; ---> IORequest beantworten
5384         rts
5386
5386         DC.W    0
5388
5388 ;-------------------------------   Ungültiges Kommando
5388
5388 FE5388  bra     FE5314               ; ---> Ungültiges Kommando
538C
538C ;-------------------------------   Signal zuweisen
538C
538C FE538C  move.l  a2,-(a7)             ; a2 retten
538E         movea.l a0,a2                ; a2 := a0 = PortBase
5390         move.b  #4,8(a2)             ; ln_Type := 4 (MsgPort)
5396         clr.b   E(a2)                ; mp_Flags := 0
539A         suba.l  a1,a1                ; a1 := 0
539C         move.l  a6,-(a7)             ; a6 retten
539E         movea.l 26(a6),a6            ; a6 := SysBase
53A2         jsr     -126(a6)             ; ---> FindTask: ThisTask
53A6         movea.l (a7)+,a6             ; a6 wiederherstellen
53A8         move.l  d0,10(a2)            ; mp_SigTask := Zeiger auf Task
53AC         lea     14(a2),a0            ; a0 -> mp_MsgList
53B0         move.l  a0,(a0)              ; Macro NewList
53B2         addq.l  #4,(a0)              ; :
53B4         clr.l   4(a0)                ; :
53B8         move.l  a0,8(a0)             ; :
53BC         moveq   #-1,d0               ; Kein bestimmtes Signalbit
53BE         move.l  a6,-(a7)             ; a6 retten
53C0         movea.l 26(a6),a6            ; a6 := SysBase
53C4         jsr     -14A(a6)             ; ---> AllocSignal
53C8         movea.l (a7)+,a6             ; a6 wiederherstellen
53CA         move.b  d0,F(a2)             ; mp_SigBit := zugewiesenes Signal
53CE         movea.l (a7)+,a2             ; a2 wiederherstellen
53D0         rts
53D2
53D2 ;-------------------------------   Signal freigeben
53D2
53D2 FE53D2  moveq   #0,d0                ; d0 := 0
```

```
53D4        move.b  F(a0),d0            ; d0 := mp_SigBit
53D8        move.l  a6,-(a7)            ; a6 retten
53DA        movea.l 26(a6),a6           ; a6 := SysBase
53DE        jsr     -150(a6)            ; ---> FreeSignal
53E2        movea.l (a7)+,a6            ; a6 wiederherstellen
53E4        rts
53E6
53E6        DC.W    0
53E8
53E8 ************************************************************************
```

Keyboard.device Datenbereich

```
-024 ************************* Keyboard.device ***************************
-024
-024          jmp    FE5220       ; ---> AbortIO
-01E          jmp    FE515C       ; ---> BeginIO
-018          jmp    FE52F4       ; ---> Extfunct
-012          jmp    FE5562       ; ---> Expunge
-00C          jmp    FE52F6       ; ---> Close
-006          jmp    FE558C       ; ---> Open
0000
0000 ;-------------------------   Library-Block
0000
0000 kbdBase DC.L    2D54         ; ln_Succ
0004          DC.L    07D4         ; ln_Pred
0008          DC.B    3            ; ln_Type (3 = Device)
0009          DC.B    3C           ; ln_Pri
000A          DC.L    FE5048       ; ln_Name -> 'keyboard.device'
000E
000E          DC.B    6            ; lib_Flags (Changed, Sumused)
000F          DC.B    0            ; lib_Pad
0010          DC.W    24           ; lib_NegSize
0012          DC.W    113          ; lib_PosSize
0014          DC.W    21           ; lib_Version
0016          DC.W    103          ; lib_Revision
0018          DC.L    FE5058       ; lib_IdString
001C          DC.L    D41E0000     ; lib_LibSum
0020          DC.W    2            ; lib_OpenCnt
0022
0022 ;-------------------------   Verschiedene Parameter
0022
0022          DC.L    0            ; SegList
0026          DC.L    676          ; SysBase
002A          DC.L    FE5506       ; CmdArray (Tabelle der IO-Kommandos)
```

```
002E          DC.L   FE553E        ; CmdFlags (Tabelle der MsgPort-Nummern)
0032          DC.W   E             ; kbd_LastCmd (Höchste Kommandonummer + 1)
0034
0034 ;-------------------------   MessagePort für ReadEvent-Requests
0034
0034    00    DC.L   0             ; ln_Succ
0038    04    DC.L   0             ; ln_Pred
003C    08    DC.B   4             ; ln_Type (4 = nt_MsgPort)
003D    09    DC.B   0             ; ln_Pri
003E    0A    DC.L   FE5048        ; ln_Name -> 'keyboard.device'
0042
0042    0E    DC.B   0             ; mp_Flags
0043    0F    DC.B   0             ; mp_SigBit
0044    10    DC.L   0             ; mp_SigTask
0048
0048    14    DC.L   43F8          ; lh_Head      \  mp_MsgList
004C    18    DC.L   0             ; lh_Tail      |
0050    1C    DC.L   43F8          ; lh_TailPred  |
0054    20    DC.B   0             ; lh_Type      |
0055    21    DC.B   0             ; lh_Pad       /
0056
0056
0056          DC.L   1A80          ; ciaa.resource
005A
005A ;------ Interrupt-Struktur für Keyboard-Interrupt 'Kode empfangen'
005A
005A    00    DC.L   0             ; ln_Succ
005E    04    DC.L   0             ; ln_Pred
0062    08    DC.B   2             ; ln_Type (2 = Interrupt)
0063    09    DC.B   0             ; ln_Pri
0064    0A    DC.L   FE5048        ; ln_Name -> 'keyboard.device'
0068    0E    DC.L   *0000         ; is_Data -> keyboard.device
006C    12    DC.L   FE571C        ; is_Code -> InputHandler
0070
0070 ;-------------------------   Puffer für Tastenkodes (RawKeyCodes)
0070
0070          DC.W   14            ; BufferTail (Lesezeiger)
0072          DC.W   14            ; BufferHead (Schreibzeiger)
0074
0074          DS.L   20            ; Puffer für 32 Kodes und Qualifier-Worte
```

```
00F4
00F4 ;-------------------------   ResetHandler-Liste
00F4
00F4          DC.L   46F6          ; lh_Head
00F8          DC.L   0             ; lh_Tail
00FC          DC.L   46F6          ; lh_TailPred
0100          DC.B   0             ; lh_Type
0101          DC.B   0             ; lh_Pad
0102          DC.W   0             ; ResetHandler-Zähler
0104          DC.B   0             ; ResetHandler-Flags
0105
0105 ;-------------------------   Qualifier-Byte
0105
0105          DC.B   0             ; Bitmuster für Shift,CapsLock,Ctrl,Alt,Amiga
0106
0106 ;-------------------------   Key-Matrix
0106
0106          DS.B   D             ; Bit-Tabelle für gedrückte Tasten (13 Bytes)
0113
0113 ************************************************************************
```

Keyboard.device Routinen

```
53E8 *************** Keyboard.device Routinen ******************************
53E8
53E8 ;---------------------------------   Initialisierung
53E8
53E8 FE53E8  movem.l aØ/a2,-(a7)          ; Register retten
53EC         lea     1Ø8(pc),aØ           ; aØ := FE54F6 -> Vektortabelle
53FØ         lea     A8(pc),a1            ; a1 := FE549A -> Tabelle f InitStruct
53F4         lea     C(pc),a2             ; a2 := FE54Ø2 -> Init-Routine
53F8         move.l  #113,dØ              ; dØ := 275 Bytes Speicherbedarf
53FE         bra     FE52A4               ; ---> Device einrichten
5402
5402 ;------ Init-Routine für MakeLibrary
5402
5402 FE5402  movem.l a6,-(a7)             ; SysBase auf Stack
5406         movea.l dØ,aØ                ; aØ := dØ -> LibBase
5408         move.l  a6,26(aØ)            ; SysBase speichern
540C         movea.l dØ,a6                ; a6 -> LibBase
540E         lea     48(a6),aØ            ; aØ -> Msg List Header
5412         move.l  aØ,(aØ)              ; Macro NewList
5414         addq.l  #4,(aØ)              ; :
5416         clr.l   4(aØ)                ; :
541A         move.l  aØ,8(aØ)             ; :
541E         lea     F4(a6),aØ            ; aØ -> ResetHandler List Header
5422         move.l  aØ,(aØ)              ; Macro NewList
5424         addq.l  #4,(aØ)              ; :
5426         clr.l   4(aØ)                ; :
542A         move.l  aØ,8(aØ)             ; :
542E         lea     3A(pc),a1            ; a1 := FE546A -> 'ciaa.resource'
5432         moveq   #Ø,dØ                ; dØ := Ø
5434         move.l  a6,-(a7)             ; a6 retten
5436         movea.l 26(a6),a6            ; a6 := SysBase
543A         jsr     -1F2(a6)             ; ---> OpenResource
```

```
543E          movea.l (a7)+,a6            ; a6 wiederherstellen
5440          move.l  d0,56(a6)           ; ResourceBase speichern
5444          beq     FE5462              ; Resource nicht geöffnet: ->
5448          bsr.s   FE5478              ; ---> Keyboard Sendesperre
544A          move.l  a6,68(a6)           ; is_Data -> keyboard.device
544E          moveq   #3,d0               ; d0 := Interrupt-Nummer
5450          lea     5A(a6),a1           ; a1 -> InterruptStructure
5454          move.l  a6,-(a7)            ; a6 retten
5456          movea.l 56(a6),a6           ; a6 := ResourceBase
545A          jsr     -6(a6)              ; ---> AddICRVector
545E          movea.l (a7)+,a6            ; a6 wiederherstellen
5460          bsr.s   FE5490              ; ---> Keyboard Sendefreigabe
5462 FE5462   move.l  a6,d0               ; d0 := a6 -> DeviceBase
5464          movem.l (a7)+,a6            ; a6 wiederherstellen
5468          rts
546A
546A FE546A   DC.B    'ciaa.resource',0
5478
5478 ;------ Keyboard-Handshake (KDAT-Leitung Low, Sendesperre)
5478
5478 FE5478   movea.l #BFE001,a0          ; a0 -> ciaa.pra
547E          moveq   #0,d0               ; d0 := 0
5480          move.b  d0,E00(a0)          ; ciaa.cra := 0
5484          move.b  d0,E00(a0)          ;
5488          move.b  #40,E00(a0)         ; SPMode := Output (KDAT = Low)
548E          rts
5490
5490 ;------ Keyboard-Handshake Ende (Sendefreigabe)
5490
5490 FE5490   move.b  #0,BFEE01           ; SPMode := Input
5498          rts
549A
549A ;--------------------------------   Tabelle für InitStruct
549A
549A FE549A   DC.W    E000                ; InitByte
549C          DC.W    8                   ; ln_Type
549E          DC.B    3                   ; = nt_Device
54A0          DC.B    0
54A0          DC.W    C000                ; InitLong
54A2          DC.W    A                   ; lib_Name
```

```
54A4        DC.L    FE5048          ; -> 'keyboard.device'
54A8        DC.W    E000            ; InitByte
54AA        DC.W    E               ; lib_Flags
54AC        DC.B    6               ; = Changed,Sumused
54AD        DC.B    0
54AE        DC.W    D000            ; InitWord
54B0        DC.W    14              ; Version
54B2        DC.W    21              ; = 33
54B4        DC.W    D000            ; InitWord
54B6        DC.W    16              ; Revision
54B8        DC.W    67              ; = 103
54BA        DC.W    C000            ; InitLong
54BC        DC.W    2A              ;
54BE        DC.L    FE5506          ; -> Kommando-Tabelle
54C2        DC.W    C000            ; InitLong
54C4        DC.W    2E
54C6        DC.L    FE553E          ; -> MsgPort-Nummern
54CA        DC.W    D000            ; InitWord
54CC        DC.W    32
54CE        DC.W    E               ; Letztes Kommando+1
54D0        DC.W    E000            ; InitByte
54D2        DC.W    62              ; ln_Type
54D4        DC.B    2               ; = nt_Interrupt
54D5        DC.B    0
54D6        DC.W    C000            ; InitLong
54D8        DC.W    64              ; ln_Name in InterruptStructure
54DA        DC.L    FE5048          ; -> 'keyboard.device'
54DE        DC.W    C000            ; InitLong
54E0        DC.W    6C              ; is_Code
54E2        DC.L    FE571C          ; -> InputHandler
54E6        DC.W    E000            ; InitByte
54E8        DC.W    3C              ; ln_Type
54EA        DC.B    4               ; = nt_MsgPort
54EB        DC.B    0
54EC        DC.W    C000            ; InitLong
54EE        DC.W    3E              ; ln_Name im MsgPort
54F0        DC.L    FE5048          ; -> 'keyboard.device'
54F4        DC.W    0
54F6
```

```
54F6 ;--------------------------------   Vektoren für Sprungliste
54F6
54F6 FE54F6  DC.W    -1                ; Relativadressen folgen
54F8         DC.W    0096              ; FE558C  Open
54FA         DC.W    FE00              ; FE52F6  Close
54FC         DC.W    006C              ; FE5562  Expunge
54FE         DC.W    FDFE              ; FE52F4  Extfunct
5500         DC.W    FC66              ; FE515C  BeginIO
5502         DC.W    FD2A              ; FE5220  AbortIO
5504         DC.W    -1                ; Endemarke
5506
5506 ;--------------------------------   Adressentabelle für IO-Kommandos
5506
5506 FE5506  DC.L    FE5314            ; $0 kbd_Invalid
550A         DC.L    FE554C            ; $1 kbd_Reset
550E         DC.L    FE5388            ; $2 kbd_Read
5512         DC.L    FE5388            ; $3 kbd_Write
5516         DC.L    FE5388            ; $4 kbd_Update
551A         DC.L    FE55A0            ; $5 kbd_Clear
551E         DC.L    FE532A            ; $6 kbd_Stop
5522         DC.L    FE533A            ; $7 kbd_Start
5526         DC.L    FE5366            ; $8 kbd_Flush
552A         DC.L    FE55C4            ; $9 kbd_ReadEvent
552E         DC.L    FE56AA            ; $A kbd_ReadMatrix
5532         DC.L    FE56DA            ; $B kbd_AddResetHandler
5536         DC.L    FE56F4            ; $C kbd_RemResetHandler
553A         DC.L    FE570A            ; $D kbd_ResetHandlerDone
553E
553E ;--------------------------------   Kommando-Flags
553E
553E FE553E  DC.B    -1,-1,-1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1,-1
554C
554C ;--------------------------------   kbd_Reset
554C
554C FE554C  bsr     FE532A            ; ---> Stop
5550         bsr     FE5478            ; ---> Keyboard Sendesperre
5554         bsr     FE5366            ; ---> Flush
5558         bsr     FE5490            ; ---> Keyboard Sendefreigabe
555C         bsr     FE533A            ; ---> Start
5560         rts
```

```
5562
5562 ;--------------------------------   kbd_Expunge
5562
5562 FE5562  tst.w   2Ø(a6)             ; Open-Zähler = Ø?
5566         bne     FE52EC             ; Nein: ->
556A         moveq   #8,dØ              ; dØ := Bitmuster für Clear SP-Int
556C         move.l  a6,-(a7)           ; a6 retten
556E         movea.l 56(a6),a6          ; a6 -> ciaa.resource
5572         jsr     -12(a6)            ; ---> AbleICR (schaltet SP-Int ab)
5576         movea.l (a7)+,a6           ; a6 wiederherstellen
5578         lea     5A(a6),a1          ; a1 -> InterruptStructure
557C         move.l  a6,-(a7)           ; a6 retten
557E         movea.l 56(a6),a6          ; a6 -> ciaa.resource
5582         jsr     -C(a6)             ; ---> RemICRVector
5586         movea.l (a7)+,a6           ; a6 wiederherstellen
5588         bra     FE52BC             ; ---> Device entfernen
558C
558C ;--------------------------------   kbd_Open
558C
558C FE558C  lea     34(a6),aØ          ; aØ -> MsgPort in keyboard.device
559Ø         move.l  aØ,18(a1)          ; als io_Unit speichern
5594         bclr.b  #3,E(a6)           ; DelExp-Flag löschen
559A         addq.w  #1,2Ø(a6)          ; Open-Zähler erhöhen
559E         rts
55AØ
55AØ ;--------------------------------   kbd_Clear
55AØ
55AØ FE55AØ  move.l  a6,-(a7)           ; a6 retten
55A2         movea.l 26(a6),a6          ; a6 := SysBase
55A6         jsr     -78(a6)            ; ---> Disable
55AA         movea.l (a7)+,a6           ; a6 wiederherstellen
55AC         clr.w   7Ø(a6)             ; BufferTail rücksetzen
55BØ         clr.w   72(a6)             ; BufferHead rücksetzen
55B4         move.l  a6,-(a7)           ; a6 retten
55B6         movea.l 26(a6),a6          ; a6 := SysBase
55BA         jsr     -7E(a6)            ; ---> Enable
55BE         movea.l (a7)+,a6           ; a6 wiederherstellen
55CØ         bra     FE56CE             ; ---> ok, IORequest beantworten
55C4
```

```
55C4 ;--------------------------------    kbd_ReadEvent
55C4
55C4 FE55C4  movem.l d2-d3/a2-a3,-(a7)   ; Register retten
55C8         movea.l a1,a3               ; a3 := a1 -> IORequest
55CA         move.l  a6,-(a7)            ; a6 retten
55CC         movea.l 26(a6),a6           ; a6 := SysBase
55D0         jsr     -78(a6)             ; ---> Disable
55D4         movea.l (a7)+,a6            ; a6 wiederherstellen
55D6         bset.b  #6,1E(a3)           ; Wiederholungssperre setzen
55DC         bne     FE5668              ; War schon gesetzt: ->
55E0         moveq   #16,d1              ; d1 := 22 = Länge des InputEvent
55E2         moveq   #0,d2               ; d2 := 0
55E4         move.l  24(a3),d3           ; d3 := io_Length
55E8         sub.l   d1,d3               ; io_Length um IE-Länge vermindern
55EA         blt     FE567A              ; Ergebnis negativ: ->
55EE         move.w  70(a6),d0           ; d0 := BufferTail
55F2         cmp.w   72(a6),d0           ; = BufferHead?
55F6         beq     FE5662              ; Ja: Puffer leer ->
55FA         move.l  a6,-(a7)            ; a6 retten
55FC         movea.l 26(a6),a6           ; a6 := SysBase
5600         jsr     -7E(a6)             ; ---> Enable
5604         movea.l (a7)+,a6            ; a6 wiederherstellen
5606         movea.l 28(a3),a2           ; a2 := io_Data
560A FE560A  move.l  a6,-(a7)            ; a6 retten
560C         movea.l 26(a6),a6           ; a6 := SysBase
5610         jsr     -78(a6)             ; ---> Disable
5614         movea.l (a7)+,a6            ; a6 wiederherstellen
5616         move.w  70(a6),d0           ; d0 := BufferTail
561A         cmp.w   72(a6),d0           ; = BufferHead?
561E         beq.s   FE5690              ; Ja: Puffer leer ->
5620         move.b  #1,4(a2)            ; ie_Class := RawKey
5626         clr.b   5(a2)               ; ie_SubClass := 0
562A         move.w  74(a6,d0.w),6(a2)   ; ie_Code eintragen
5630         move.w  76(a6,d0.w),8(a2)   ; ie_Qualifier eintragen
5636         clr.l   A(a2)               ; ie_EventAddress löschen
563A         clr.w   E(a2)               ; ie_TimeStamp, 1. Wort
563E         addq.w  #4,d0               ; BufferTail um 4 erhöhen
5640         adda.l  d1,a2               ; a2 -> Nächster InputEvent
5642         move.l  a2,-16(a2)          ; a2 als NextEvent eintragen
5646         andi.w  #7F,d0              ; BufferTail modulo 128
```

```
564A          move.w  d0,70(a6)          ; als neuen BufferTail eintragen
564E          add.l   d1,d2              ; d2 um InputEvent-Länge erhöhen
5650          move.l  a6,-(a7)           ; a6 retten
5652          movea.l 26(a6),a6          ; a6 := SysBase
5656          jsr     -7E(a6)            ; ---> Enable
565A          movea.l (a7)+,a6           ; a6 wiederherstellen
565C          sub.l   d1,d3              ; io_Length um IE-Länge vermindern
565E          blt.s   FE5690             ; Ergebnis negativ: ->
5660          bra.s   FE560A             ; ---> Loop
5662
5662 FE5662   andi.b  #BE,1E(a3)         ; Wiederholungssperre rücksetzen
5668 FE5668   move.l  a6,-(a7)           ; a6 retten
566A          movea.l 26(a6),a6          ; a6 := SysBase
566E          jsr     -7E(a6)            ; ---> Enable
5672          movea.l (a7)+,a6           ; a6 wiederherstellen
5674 FE5674   movem.l (a7)+,d2-d3/a2-a3  ; Register wiederherstellen
5678          rts
567A
567A FE567A   move.l  a6,-(a7)           ; a6 retten
567C          movea.l 26(a6),a6          ; a6 := SysBase
5680          jsr     -7E(a6)            ; ---> Enable
5684          movea.l (a7)+,a6           ; a6 wiederherstellen
5686          moveq   #-4,d0             ; d0 := ioerr_BadLength
5688          movea.l a3,a1              ; a1 := a3 -> IORequest
568A          bsr     FE5226             ; ---> IORequest beantworten
568E          bra.s   FE569C             ; --->
5690
5690 FE5690   clr.l   -16(a2)            ; Letzten ie_NextEvent löschen
5694          move.l  d2,20(a3)          ; io_Actual eintragen
5698          movea.l a3,a1              ; a1 := a3 -> IORequest
569A          bsr.s   FE56CE             ; ---> IORequest beantworten
569C FE569C   movea.l 48(a6),a1          ; a1 -> Request-Liste
56A0          tst.l   (a1)               ; Ende der Liste?
56A2          beq.s   FE5674             ; Ja: Abschluß ->
56A4          bsr     FE55C4             ; ---> ReadEvent
56A8          bra.s   FE5674             ; ---> Abschluß
56AA
56AA ;--------------------------------    kbd_ReadMatrix
56AA
56AA FE56AA   moveq   #D,d0              ; d0 := 13 (Bytes in der Matrix)
```

```
56AC          cmp.l   24(a1),dØ           ; io_Length = 13?
56BØ          bne.s   FE56D4              ; Nein: Fehler ->
56B2          movea.l 28(a1),aØ           ; aØ -> io_Data
56B6          move.l  a1,-(a7)            ; a1 retten
56B8          lea     1Ø6(a6),a1          ; a1 -> Matrix-Tabelle
56BC          subq.w  #1,dØ               ; dØ := 12 = Bytezähler
56BE FE56BE   move.b  (a1)+,(aØ)+         ; Matrixbyte kopieren
56CØ          dbra    dØ,FE56BE           ; dØ-mal wiederholen ->
56C4          movea.l (a7)+,a1            ; a1 wiederherstellen
56C6          move.l  #D,2Ø(a1)           ; io_Actual := 13
56CE FE56CE   moveq   #Ø,dØ               ; dØ := ok-Kode
56DØ          bra     FE5226              ; ---> IORequest beantworten
56D4
56D4 FE56D4   moveq   #-4,dØ              ; dØ := ioerr_BadLength
56D6          bra     FE5226              ; ---> IORequest beantworten
56DA
56DA ;--------------------------------     kbd_AddResetHandler
56DA
56DA FE56DA   move.l  a1,-(a7)            ; a1 retten
56DC          movea.l 28(a1),a1           ; a1 -> ResetHandler
56EØ          lea     F4(a6),aØ           ; aØ -> ResetHandler List Header
56E4          move.l  a6,-(a7)            ; a6 retten
56E6          movea.l 26(a6),a6           ; a6 := SysBase
56EA          jsr     -1ØE(a6)            ; ---> Enqueue
56EE          movea.l (a7)+,a6            ; a6 wiederherstellen
56FØ          movea.l (a7)+,a1            ; a1 wiederherstellen
56F2          bra.s   FE56CE              ; ---> ok, IORequest beantworten
56F4
56F4 ;--------------------------------     kbd_RemResetHandler
56F4
56F4 FE56F4   move.l  a1,-(a7)            ; a1 retten
56F6          movea.l 28(a1),a1           ; a1 -> ResetHandler
56FA          movea.l (a1),aØ             ; Macro Remove
56FC          movea.l 4(a1),a1            ; :
57ØØ          move.l  aØ,(a1)             ; :
5702          move.l  a1,4(aØ)            ; :
5706          movea.l (a7)+,a1            ; a1 wiederherstellen
5708          bra.s   FE56CE              ; ---> ok, IORequest beantworten
57ØA
```

```
570A ;--------------------------------    kbd_ResetHandlerDone
570A
570A FE570A  subq.w  #1,102(a6)          ; ResetHandler-Zähler erniedrigen
570E         bne.s   FE5718              ; noch nicht Null: ->
5710         andi.b  #BF,BFEE01          ; Keyboard abschalten
5718 FE5718  bra.s   FE56CE              ; ---> ok, IORequest beantworten
571A
571A         DC.W    0
571C
571C ;--------------------------------    kbd_InputHandler
571C
571C FE571C  movem.l d2-d4/a6,-(a7)      ; Register retten
5720         movea.l a1,a6               ; a6 := a1 -> keyboard.device
5722         movea.l #BFE001,a0          ; a0 -> ciaa
5728         moveq   #0,d0               ; d0 := 0
572A         move.b  C00(a0),d0          ; d0 := SDR: Datenbyte vom Keyboard
572E         ori.b   #40,E00(a0)         ; Keyboard abschalten
5734         moveq   #0,d4               ; d4 := 0
5736         not.b   d0                  ; Keyboard-Datenbyte invertieren
5738         ror.b   #1,d0               ; und um 1 Bit nach rechts rotieren
573A         cmpi.b  #78,d0              ; Reset-Kode? (Entstehung unklar!)
573E         bne.s   FE577A              ; Nein: ->
5740         bset.b  #0,104(a6)          ; Verzögerungsflag setzen
5746         beq.s   FE5780              ; War nicht gesetzt: ->
5748         bset.b  #1,104(a6)          ; Wiederholungssperre setzen
574E         bne     FE584E              ; War schon gesetzt: ->
5752         movea.l F4(a6),a1           ; a1 := lh_Head (ResetHandler Liste)
5756         tst.l   (a1)                ; Liste leer?
5758         beq     FE5846              ; Ja: ->
575C FE575C  move.l  (a1),d2             ; d2 -> Nächster ResetHandler
575E         beq     FE584E              ; Ende der Liste: ->
5762         addq.w  #1,102(a6)          ; ResetHandler-Zähler erhöhen
5766         move.l  a6,-(a7)            ; a6 retten
5768         movea.l 26(a6),a6           ; a6 := SysBase
576C         jsr     -B4(a6)             ; ---> Cause
5770         movea.l (a7)+,a6            ; a6 wiederherstellen
5772         movea.l d2,a1               ; a1 := d2 -> Nächster ResetHandler
5774         bra.s   FE575C              ; ---> Loop
5776
5776         bne     FE584E
```

```
577A FE577A  bclr.b  #0,104(a6)          ; Reset-Verzögerungsflag löschen
578Ø FE578Ø  move.w  dØ,d1               ; d1 := dØ = Tastenkode
5782         andi.w  #7F,d1              ; Bit 7 löschen
5786         cmpi.w  #67,d1              ; Kode > $67?
578A         bhi     FE57EC              ; Ja: ->
578E         move.w  d1,d2               ; d2 := d1 = Tastenkode
5790         andi.w  #7,d2               ; d2 := Tastenkode modulo 8
5794         lsr.w   #3,d1               ; d1 := Tastenkode \ 8
5796         lea     106(a6),aØ          ; aØ -> Bit-Matrix          _
579A         btst.l  #7,dØ               ; Taste losgelassen?        |
579E         bne.s   FE57BØ              ; Ja: ->                    | Qualifier-Bits
57AØ         bset.b  d2,Ø(aØ,d1.w)       ; Tastenbit setzen          |
57A4         cmpi.w  #C,d1               ; Qualifier-Taste?          | Ø  LShift
57A8         bne.s   FE57DØ              ; Nein: ->                  | 1  RShift
57AA         bset.b  d2,105(a6)          ; Qualifier-Bit setzen      | 2  CapsLock
57AE         bra.s   FE57DØ              ; --->                      | 3  Ctrl
57BØ                                                                 | 4  LAlt
57BØ FE57BØ  bclr.b  d2,Ø(aØ,d1.w)       ; Tastenbit löschen         | 5  RAlt
57B4         cmpi.w  #C,d1               ; Qualifier-Taste?          | 6  LAmiga
57B8         bne.s   FE57DØ              ; Nein: ->                  | 7  RAmiga
57BA         bclr.b  d2,105(a6)          ; Qualifier-Bit löschen     | 8  NumKeyPad
57BE         bra.s   FE57DØ              ; --->                      |_
57CØ
57CØ FE57CØ  DC.B    Ø,8Ø,Ø,EØ,Ø,EØ,Ø,FØ,8,4,Ø,Ø,Ø,Ø,Ø,Ø ; NumKeyPad-Tabelle
57DØ
57DØ FE57DØ  move.w  dØ,d1               ; d1 := Tastenkode
57D2         andi.w  #7F,d1              ; Auf/Ab-Bit löschen
57D6         move.w  d1,d2               ; d2 := d1 = Tastenkode
57D8         andi.w  #7,d2               ; d2 := Tastenkode modulo 8
57DC         lsr.w   #3,d1               ; d1 := Tastenkode \ 8
57DE         lea     -2Ø(pc),aØ          ; aØ := FE57CØ -> NumKeyPad-Tabelle
57E2         btst.b  d2,Ø(aØ,d1.w)       ; NumKeyPad-Taste?
57E6         beq.s   FE57EC              ; Nein: ->
57E8         ori.w   #1ØØ,d4             ; Bit 8 im Qualifier-Wort setzen
57EC FE57EC  move.l  a6,-(a7)            ; a6 retten
57EE         movea.l 26(a6),a6           ; a6 := SysBase
57F2         jsr     -78(a6)             ; ---> Disable
57F6         movea.l (a7)+,a6            ; a6 wiederherstellen
57F8         move.w  72(a6),d2           ; d2 -> BufferHead
57FC         move.w  d2,d3               ; d3 := d2 -> BufferHead
```

```
57FE          addq.w  #4,d2               ; BufferHead um 4 erhöhen
5800          andi.w  #7F,d2              ; modulo 128
5804          cmp.w   70(a6),d2           ; BufferTail erreicht?
5808          bne.s   FE5816              ; Nein: ->
580A          move.w  d3,d2               ; d2 wiederherstellen
580C          move.w  #7D,d0              ; Tastenkode durch $7D ersetzen
5810          subq.w  #4,d3               ; BufferHead um 4 erniedrigen
5812          andi.w  #7F,d3              ; modulo 128
5816 FE5816   move.w  d0,74(a6,d3.w)      ; Tastenkode in Puffer speichern
581A          or.b    105(a6),d4          ; Qualifier-Bit einodern
581E          move.w  d4,76(a6,d3.w)      ; Qualifier-Wort in Puffer speichern
5822          move.w  d2,72(a6)           ; BufferHead speichern
5826          move.l  a6,-(a7)            ; a6 retten
5828          movea.l 26(a6),a6           ; a6 := SysBase
582C          jsr     -7E(a6)             ; ---> Enable
5830          movea.l (a7)+,a6            ; a6 wiederherstellen
5832          btst.b  #0,3D(a6)           ; Stop-Flag gesetzt?
5838          bne.s   FE5846              ; Ja: ->
583A          movea.l 48(a6),a1           ; a1 -> ReadRequest-Liste
583E          tst.l   (a1)                ; Liste leer?
5840          beq.s   FE5846              ; Ja: ->
5842          bsr     FE55C4              ; ---> ReadEvent
5846 FE5846   andi.b  #BF,BFEE01          ; Keyboard einschalten
584E FE584E   movem.l (a7)+,d2-d4/a6      ; Register wiederherstellen
5852          rts
5854
5854 ************************************************************************
```

Gameport.device Datenbereich

```
-024 ************************** Gameport.device ****************************
-024
-024          jmp    FE5220       ; ---> AbortIO
-01E          jmp    FE515C       ; ---> BeginIO
-018          jmp    FE52F4       ; ---> Extfunct
-012          jmp    FE59BA       ; ---> Expunge
-00C          jmp    FE52F6       ; ---> Close
-006          jmp    FE59D8       ; ---> Open
0000
0000 ;-------------------------   Library-Block
0000
0000 gpdBase DC.L    C022BE       ; ln_Succ
0004         DC.L    C01FFC       ; ln_Pred
0008         DC.B    3            ; ln_Type (3 = Device)
0009         DC.B    0            ; ln_Pri
000A         DC.L    FE5094       ; ln_Name -> 'gameport.device'
000E
000E         DC.B    6            ; dev_Flags (Changed, Sumused)
000F         DC.B    0            ; dev_Pad
0010         DC.W    24           ; dev_NegSize
0012         DC.W    14D          ; dev_PosSize
0014         DC.W    21           ; dev_Version
0016         DC.W    67           ; dev_Revision
0018         DC.L    0            ; dev_IdString
001C         DC.L    DCC20000     ; dev_LibSum
0020         DC.W    1            ; dev_OpenCnt
0022
0022 ;-------------------------   Verschiedene Parameter
0022
0022         DC.L    0            ; SegList
0026         DC.L    C00276       ; SysBase
002A         DC.L    FE5974       ; CmdArray
```

```
002E          DC.L   FE59AC        ; CmdFlags
0032          DC.W   E             ; NumCmds+1
0034
0034          DC.L   C01632        ; potgo.resource
0038
0038 ;-------------------------   GamePort 0
0038
0038 ;------ MessagePort für ReadEvent-Requests
0038
0038    00    DC.L   0             ; ln_Succ
003C    04    DC.L   0             ; ln_Pred
0040    08    DC.B   4             ; ln_Type (4 = MsgPort)
0041    09    DC.B   0             ; ln_Pri
0042    0A    DC.L   FE5094        ; ln_Name -> 'gameport.device'
0046
0046    0E    DC.B   0             ; mp_Flags
0047    0F    DC.B   0             ; mp_SigBit
0048    10    DC.L   0             ; mp_SigTask
004C
004C    14    DC.L   C03792        ; lh_Head      \  mp_MsgList
0050    18    DC.L   0             ; lh_Tail      |
0054    1C    DC.L   C03792        ; lh_TailPred  |
0058    20    DC.B   0             ; lh_Type      |
0059    21    DC.B   0             ; lh_Pad       /
005A
005A ;------ Input-Puffer
005A
005A    22    DC.W   20            ; BufferTail
005C    24    DC.W   20            ; BufferHead
005E
005E    26    DS.L   10            ; Puffer (64 Bytes)
009E
009E ;------ Triggerparameter
009E
009E    66    DC.W   3             ; gpt_Keys (Bit 0: Downkeys, Bit 1: Upkeys)
00A0    68    DC.W   0             ; gpt_Timeout
00A2    6A    DC.W   1             ; gpt_XDelta
00A4    6C    DC.W   1             ; gpt_YDelta
00A6
```

```
00A6 ;------ Input-Variablen
00A6
00A6    6E   DC.W   0            ; Maustasten-Status
00A8    70   DC.W   -1           ; Zeitzähler (+1 bei jedem Interrupt)
00AA    72   DC.W   0            ; X-Wert
00AC    74   DC.W   0            ; Y-Wert
00AE    76   DC.B   0            ; Y-Wert bei Triggerung
00AF    77   DC.B   0            ; X-Wert bei Triggerung
00B0    78   DC.W   2FF1         ; Mauszählerstand (X: 0,..,7; Y: 8,..,15)
00B2    7A   DC.B   8            ; Flags (1: SetCType, 2: Port1, 3: RelJoyStick)
00B3    7B   DC.B   1            ; ControllerType (1 = Maus)
00B4
00B4 ;-------------------------   GamePort 1
00B4
00B4 ;------ MessagePort für ReadEvent-Requests
00B4
00B4    00   DC.L   0            ; ln_Succ
00B8    04   DC.L   0            ; ln_Pred
00BC    08   DC.B   4            ; ln_Type (4 = MsgPort)
00BD    09   DC.B   0            ; ln_Pri
00BE    0A   DC.L   FE5094       ; ln_Name -> 'gameport.device'
00C2
00C2    0E   DC.B   0            ; mp_Flags
00C3    0F   DC.B   0            ; mp_SigBit
00C4    10   DC.L   0            ; mp_SigTask
00C8
00C8    14   DC.L   *00CC        ; lh_Head      \  mp_MsgList
00CC    18   DC.L   0            ; lh_Tail      |
00D0    1C   DC.L   *00C8        ; lh_TailPred  |
00D4    20   DC.B   0            ; lh_Type      |
00D5    21   DC.B   0            ; lh_Pad       /
00D6
00D6 ;------ Input Buffer
00D6
00D6    22   DC.W   0            ; BufferTail
00D8    24   DC.W   0            ; BufferHead
00DA
00DA    26   DS.L   10           ; Puffer (64 Bytes)
011A
```

```
011A ;------ Triggerparameter
011A
011A    66   DC.W   0            ; gpt_Keys (Bit 0: Downkeys, Bit 1: Upkeys)
011C    68   DC.W   0            ; gpt_Timeout
011E    6A   DC.W   0            ; gpt_XDelta
0120    6C   DC.W   0            ; gpt_YDelta
0122
0122 ;------ Input Variablen
0122
0122    6E   DC.W   0            ; Maustasten-Status
0124    70   DC.W   -1           ; Zeitzähler (+1 bei jedem Interrupt)
0126    72   DC.W   0            ; X-Wert
0128    74   DC.W   0            ; Y-Wert
012A    76   DC.B   0            ; Y-Wert bei Triggerung
012B    77   DC.B   0            ; X-Wert bei Triggerung
012C    78   DC.W   0            ; Mauszählerstand (X: 0,..,7; Y: 8,..,15)
012E    7A   DC.B   4            ; Flags (1: SetCType, 2: Port1, 3: RelJoyStick)
012F    7B   DC.B   0            ; ControllerType
0130
0130 ;-------------------------   Interrupt-Struktur für Bildwechsel-Interrupt
0130
0130         DC.L   C02360       ; ln_Succ
0134         DC.L   C01E88       ; ln_Pred
0138         DC.B   2            ; ln_Type (2 = Interrupt)
0139         DC.B   0            ; ln_Pri
013A         DC.L   FE5094       ; ln_Name -> 'gameport.device'
013E         DC.L   *0000        ; is_Data -> gameport.device
0142         DC.L   FE5CC8       ; is_Code -> gpd_InputHandler
0146
0146 ;-------------------------   PotBits
0146
0146         DC.W   0F00         ; Letzte PotBit-Zuordnung
0148         DC.W   0F00         ; zugeordnete PotBits (Maske für WritePotgo)
014A         DC.W   0F00         ; aktuelle PotBits
014C
014C         DC.B   0
014D
014D ************************************************************************
```

Gameport.device Routinen

```
5854 *************** Gameport.device Routinen ******************************
5854
5854 ;--------------------------------   Initialisierung
5854
5854 FE5854  movem.l aØ/a2,-(a7)          ; Register retten
5858         lea     1ØA(pc),aØ           ; aØ := FE5964 -> Vektortabelle
585C         lea     9Ø(pc),a1            ; a1 := FE58EE -> Tabelle f InitStruct
586Ø         lea     C(pc),a2             ; a2 := FE586E -> Init-Routine
5864         move.l  #14D,dØ              ; dØ := 333 Bytes Speicherbedarf
586A         bra     FE52A4               ; ---> Device einrichten
586E
586E ;------ Init-Routine für MakeLibrary
586E
586E FE586E  movem.l a6,-(a7)             ; a6 retten
5872         movea.l dØ,aØ                ; aØ := dØ = DevBase
5874         move.l  a6,26(aØ)            ; SysBase speichern
5878         movea.l dØ,a6                ; a6 := dØ = DevBase
587A         lea     62(pc),a1            ; a1 := FE58DE -> 'potgo.resource'
587E         move.l  a6,-(a7)             ; a6 retten
588Ø         movea.l 26(a6),a6            ; a6 := SysBase
5884         jsr     -1F2(a6)             ; ---> OpenResource
5888         movea.l (a7)+,a6             ; a6 wiederherstellen
588A         move.l  dØ,34(a6)            ; PotgoBase speichern
588E         lea     4C(a6),aØ            ; aØ -> MessageList für Port 1
5892         move.l  aØ,(aØ)              ; Macro NewList
5894         addq.l  #4,(aØ)              ; :
5896         clr.l   4(aØ)                ; :
589A         move.l  aØ,8(aØ)             ; :
589E         lea     C8(a6),aØ            ; aØ -> MessageList für Port 2
58A2         move.l  aØ,(aØ)              ; Macro NewList
58A4         addq.l  #4,(aØ)              ; :
58A6         clr.l   4(aØ)                ; :
```

```
58AA          move.l  aØ,8(aØ)            ; :
58AE          move.l  a6,13E(a6)           ; is_Data -> gameport.device
58B2          moveq   #5,dØ                ; dØ := Interrupt Nummer
58B4          lea     13Ø(a6),a1           ; a1 -> Interrupt
58B8          move.l  a6,-(a7)             ; a6 retten
58BA          movea.l 26(a6),a6            ; a6 := SysBase
58BE          jsr     -A8(a6)              ; ---> AddIntServer
58C2          movea.l (a7)+,a6             ; a6 wiederherstellen
58C4          movea.l #BFEØØ1,aØ           ; aØ := ciaaBase
58CA          andi.b  #3F,2ØØ(aØ)          ; ciaa.ddra: Bits 6,7 = Input
58DØ          ori.b   #CØ,Ø(aØ)            ; ciaa.pra:  Bits 6,7 setzen
58D6          move.l  a6,dØ                ; dØ := gptBase
58D8          movem.l (a7)+,a6             ; a6 wiederherstellen
58DC          rts
58DE
58DE FE58DE   DC.B    'potgo.resource',Ø,Ø
58EE
58EE ;--------------------------------   Tabelle für InitStruct
58EE
58EE FE58EE   DC.W    EØØØ                 ; InitByte
58FØ          DC.W    8                    ; ln_Type
58F2          DC.B    3                    ; = nt_Device
58F3          DC.B    Ø
58F4          DC.W    CØØØ                 ; InitLong
58F6          DC.W    A                    ; lib_Name
58F8          DC.L    FE5Ø94               ; -> 'gameport.device'
58FC          DC.W    EØØØ                 ; InitByte
58FE          DC.W    E                    ; lib_Flags
59ØØ          DC.B    6                    ; = Changed,Sumused
59Ø1          DC.B    Ø
59Ø2          DC.W    DØØØ                 ; InitWord
59Ø4          DC.W    14                   ; Version
59Ø6          DC.W    21                   ; = 33
59Ø8          DC.W    DØØØ                 ; InitWord
59ØA          DC.W    16                   ; Revision
59ØC          DC.W    67                   ; = 1Ø3
59ØE          DC.W    CØØØ                 ; InitLong
591Ø          DC.W    2A
5912          DC.L    FE5974               ; -> Kommando-Tabelle
5916          DC.W    CØØØ                 ; InitLong
```

```
5918        DC.W    2E
591A        DC.L    FE59AC              ; -> MsgPort-Nummern
591E        DC.W    D000                ; InitWord
5920        DC.W    32
5922        DC.W    E                   ; Letztes Kommando+1
5924        DC.W    E000                ; InitByte
5926        DC.W    138                 ; ln_Type
5928        DC.B    2                   ; = nt_Interrupt
5929        DC.B    0
592A        DC.W    E000                ; InitByte
592C        DC.W    139                 ; ln_Pri
592E        DC.B    0                   ; = 0
592F        DC.B    0
5930        DC.W    C000                ; InitLong
5932        DC.W    13A                 ; ln_Name in InterruptStructure
5934        DC.L    FE5094              ; -> 'gameport.device'
5938        DC.W    C000                ; InitLong
593A        DC.W    142                 ; is_Code
593C        DC.L    FE5CC8              ; -> Interrupt-Routine
5940        DC.W    E000                ; InitByte
5942        DC.W    40                  ; ln_Type
5944        DC.B    4                   ; = nt_MsgPort
5945        DC.B    0
5946        DC.W    C000                ; InitLong
5948        DC.W    42                  ; ln_Name im MsgPort
594A        DC.L    FE5094              ; -> 'gameport.device'
594E        DC.W    E000                ; InitByte
5950        DC.W    BC                  ; ln_Type
5952        DC.B    4                   ; = nt_MsgPort
5953        DC.B    0
5954        DC.W    C000                ; InitLong
5956        DC.W    BE                  ; ln_Name im MsgPort
5958        DC.L    FE5094              ; -> 'gameport.device'
595C        DC.W    E000                ; InitByte
595E        DC.W    12E                 ; Flags
5960        DC.B    4                   ;
5961        DC.B    0
5962        DC.W    0
5964
```

```
5964 ;--------------------------------   Vektoren für Sprungliste
5964
5964 FE5964  DC.W    -1                 ; Relativadressen folgen
5966         DC.W    ØØ74               ; FE59D8  Open
5968         DC.W    F992               ; FE52F6  Close
596A         DC.W    ØØ56               ; FE59BA  Expunge
596C         DC.W    F99Ø               ; FE52F4  Extfunct
596E         DC.W    F7F8               ; FE515C  BeginIO
5970         DC.W    F8BC               ; FE522Ø  AbortIO
5972         DC.W    -1                 ; Endemarke
5974
5974 ;--------------------------------   Adressentabelle für IO-Kommandos
5974
5974 FE5974  DC.L    FE5314             ; $Ø gpd_Invalid
5978         DC.L    FE531C             ; $1 gpd_Reset
597C         DC.L    FE5388             ; $2 gpd_Read
598Ø         DC.L    FE5388             ; $3 gpd_Write
5984         DC.L    FE5388             ; $4 gpd_Update
5988         DC.L    FE5AØ4             ; $5 gpd_Clear
598C         DC.L    FE532A             ; $6 gpd_Stop
599Ø         DC.L    FE533A             ; $7 gpd_Start
5994         DC.L    FE5366             ; $8 gpd_Flush
5998         DC.L    FE5A2A             ; $9 gpd_ReadEvent
599C         DC.L    FE5B36             ; $A gpd_AskCType
59AØ         DC.L    FE5B5A             ; $B gpd_SetCType
59A4         DC.L    FE5C62             ; $C gpd_AskTrigger
59A8         DC.L    FE5C92             ; $D gpd_SetTrigger
59AC
59AC ;--------------------------------   Kommando-Flags
59AC
59AC FE59AC  DC.B    -1,-1,-1,-1,-1,-1,-1,-1,-1,Ø,-1,-1,-1,-1
59BA
59BA ;--------------------------------   gpd_Expunge
59BA
59BA FE59BA  tst.w   2Ø(a6)             ; Open-Zähler = Ø?
59BE         bne     FE52EC             ; Nein: ->
59C2         lea     13Ø(a6),a1         ; a1 -> Interrupt
59C6         moveq   #5,dØ              ; dØ := Interrupt Number
59C8         move.l  a6,-(a7)           ; a6 retten
59CA         movea.l 26(a6),a6          ; a6 := SysBase
```

```
59CE          jsr     -AE(a6)             ; ---> RemIntServer
59D2          movea.l (a7)+,a6            ; a6 wiederherstellen
59D4          bra     FE52BC              ; ---> Device entfernen
59D8
59D8 ;--------------------------------   gpd_Open
59D8
59D8 FE59D8   tst.l   dØ                  ; UnitNumber
59DA          bne.s   FE59E2              ; <> Ø: ->
59DC          lea     38(a6),aØ           ; aØ -> MsgPort 1
59EØ          bra.s   FE59EA              ; --->
59E2
59E2 FE59E2   subq.w  #1,dØ               ; UnitNumber - 1
59E4          bne.s   FE59FA              ; <> Ø: Fehler ->
59E6          lea     B4(a6),aØ           ; aØ -> MsgPort 2
59EA FE59EA   move.l  aØ,18(a1)           ; aØ als io_Unit speichern
59EE          bclr.b  #3,E(a6)            ; DelExp-Flag löschen
59F4          addq.w  #1,2Ø(a6)           ; Open-Zähler erhöhen
59F8 FE59F8   rts
59FA
59FA FE59FA   moveq   #-1,dØ              ; dØ := -1
59FC          move.b  dØ,1F(a1)           ; als io_Error speichern
5AØØ          bra.s   FE59F8              ; --->
5AØ2
5AØ2          DC.W    Ø
5AØ4
5AØ4 ;--------------------------------   gpd_Clear
5AØ4
5AØ4 FE5AØ4   movea.l 18(a1),aØ          ; aØ -> io_Unit
5AØ8          move.l  a6,-(a7)            ; a6 retten
5AØA          movea.l 26(a6),a6           ; a6 := SysBase
5AØE          jsr     -78(a6)             ; ---> Disable
5A12          movea.l (a7)+,a6            ; a6 wiederherstellen
5A14          clr.w   22(aØ)              ; BufferTail löschen
5A18          clr.w   24(aØ)              ; BufferHead löschen
5A1C          move.l  a6,-(a7)            ; a6 retten
5A1E          movea.l 26(a6),a6           ; a6 := SysBase
5A22          jsr     -7E(a6)             ; ---> Enable
5A26          movea.l (a7)+,a6            ; a6 wiederherstellen
5A28          rts
5A2A
```

```
5A2A ;--------------------------------  gpd_ReadEvent
5A2A
5A2A FE5A2A  movem.l d2-d4/a2-a4,-(a7)   ; Register retten
5A2E         movea.l a1,a3               ; a3 := a1 -> IORequest
5A3Ø         movea.l 18(a3),a4           ; a4 := io_Unit
5A34         move.l  a6,-(a7)            ; a6 retten
5A36         movea.l 26(a6),a6           ; a6 := SysBase
5A3A         jsr     -78(a6)             ; ---> Disable
5A3E         movea.l (a7)+,a6            ; a6 wiederherstellen
5A4Ø         bset.b  #6,1E(a3)           ; Wiederholsperre setzen
5A46         bne     FE5AFØ              ; War schon gesetzt: ->
5A4A         moveq   #16,d1              ; d1 := 16 = Länge des InputEvent
5A4C         moveq   #Ø,d2               ; d2 := Ø = Anfangswert von io_Actual
5A4E         move.l  24(a3),d3           ; d3 := io_Length
5A52         sub.l   d1,d3               ; Pufferlänge - InputEvent-Länge
5A54         blt     FE5BØ2              ; < Ø: Puffer zu klein ->
5A58         move.w  22(a4),dØ           ; dØ := BufferTail
5A5C         cmp.w   24(a4),dØ           ; = BufferHead?
5A6Ø         beq     FE5AEA              ; ja: Puffer leer ->
5A64         move.l  a6,-(a7)            ; a6 retten
5A66         movea.l 26(a6),a6           ; a6 := SysBase
5A6A         jsr     -7E(a6)             ; ---> Enable
5A6E         movea.l (a7)+,a6            ; a6 wiederherstellen
5A7Ø         movea.l 28(a3),a2           ; a2 := io_Data
5A74 FE5A74  move.l  a6,-(a7)            ; a6 retten
5A76         movea.l 26(a6),a6           ; a6 := SysBase
5A7A         jsr     -78(a6)             ; ---> Disable
5A7E         movea.l (a7)+,a6            ; a6 wiederherstellen
5A8Ø         move.w  22(a4),dØ           ; dØ := BufferTail
5A84         cmp.w   24(a4),dØ           ; = BufferHead?
5A88         beq     FE5B18              ; ja: Puffer leer ->
5A8C         move.b  #2,4(a2)            ; ie_Class := ieclass_RawMouse
5A92         btst.b  #2,7A(a4)           ; Linker Port?
5A98         beq.s   FE5AA2              ; Ja: ->
5A9A         move.b  #1,5(a2)            ; ie_SubClass := 1 (rechter Port)
5AAØ         bra.s   FE5AA6              ; --->
5AA2
5AA2 FE5AA2  clr.b   5(a2)               ; ie_SubClass := Ø (linker Port)
5AA6 FE5AA6  moveq   #Ø,d4               ; d4 := Ø
5AA8         move.b  26(a4,dØ.w),d4      ; d4 := Gameport Button Report
```

```
5AAC          move.w  d4,6(a2)            ; ie_Code := d4
5ABØ          move.b  27(a4,dØ.w),d4      ; d4 := Relative & Button Bits
5AB4          lsl.w   #8,d4               ; um 8 Bits linksverschieben
5AB6          move.w  d4,8(a2)            ; ie_Qualifier := d4
5ABA          move.l  28(a4,dØ.w),A(a2)   ; ie_X und ie_Y kopieren
5ACØ          move.l  2C(a4,dØ.w),E(a2)   ; Ticks kopieren (warum .l? vgl. 5EB8!)
5AC6          addq.w  #8,dØ               ; BufferTail um 8 erhöhen
5AC8          adda.l  d1,a2               ; Zeiger auf InputEvent um 22 erhöhen
5ACA          move.l  a2,-16(a2)          ; und als ie_NextEvent eintragen
5ACE          andi.w  #3F,dØ              ; BufferTail mod 64
5AD2          move.w  dØ,22(a4)           ; als neuen BufferTail eintragen
5AD6          add.l   d1,d2               ; io_Actual um InpEvent-Länge erhöhen
5AD8          move.l  a6,-(a7)            ; a6 retten
5ADA          movea.l 26(a6),a6           ; a6 := SysBase
5ADE          jsr     -7E(a6)             ; ---> Enable
5AE2          movea.l (a7)+,a6            ; a6 wiederherstellen
5AE4          sub.l   d1,d3               ; Pufferrest - InpEvent-Länge
5AE6          blt.s   FE5B18              ; < Ø: Kein Platz mehr für InpEvent ->
5AE8          bra.s   FE5A74              ; ---> Loop
5AEA
5AEA FE5AEA   andi.b  #BE,1E(a3)          ; Wiederholsperre und io_Quick löschen
5AFØ FE5AFØ   move.l  a6,-(a7)            ; a6 retten
5AF2          movea.l 26(a6),a6           ; a6 := SysBase
5AF6          jsr     -7E(a6)             ; ---> Enable
5AFA          movea.l (a7)+,a6            ; a6 wiederherstellen
5AFC FE5AFC   movem.l (a7)+,d2-d4/a2-a4   ; Register wiederherstellen
5BØØ          rts
5BØ2
5BØ2 ;------ Fehlerausgang
5BØ2
5BØ2 FE5BØ2   move.l  a6,-(a7)            ; a6 retten
5BØ4          movea.l 26(a6),a6           ; a6 := SysBase
5BØ8          jsr     -7E(a6)             ; ---> Enable
5BØC          movea.l (a7)+,a6            ; a6 wiederherstellen
5BØE          moveq   #-4,dØ              ; dØ := ioerr_BadLength
5B1Ø          movea.l a3,a1               ; a1 := a3 -> io_Request
5B12          bsr     FE5226              ; ---> IORequest beantworten
5B16          bra.s   FE5B28              ; --->
5B18
```

```
5B18 ;------ Ausgang nach Lesen von Events
5B18
5B18 FE5B18  clr.l   -16(a2)            ; io_NextEvent := 0
5B1C         move.l  d2,20(a3)          ; io_Actual := Länge aller Events
5B20         movea.l a3,a1              ; a1 := a3 -> IORequest
5B22         moveq   #0,d0              ; d0 := ok-Kode
5B24         bsr     FE5226             ; ---> IORequest beantworten
5B28 FE5B28  movea.l 14(a4),a1          ; a1 := MsgList.lh_Head
5B2C         tst.l   (a1)               ; Ende der Liste?
5B2E         beq.s   FE5AFC             ; ja: Fertig ->
5B30         bsr     FE5A2A             ; ---> gpd_ReadEvent
5B34         bra.s   FE5AFC             ; ---> Abschluß
5B36
5B36 ;-------------------------------   gpd_AskCType
5B36
5B36 FE5B36  move.l  24(a1),d0          ; d0 := io_Length
5B3A         subq.w  #1,d0              ; - 1
5B3C         bcs     FE5CC2             ; < 0: Fehler ->
5B40         movea.l 18(a1),a0          ; a0 := io_Unit
5B44         move.b  7B(a0),d1          ; d1 := CType
5B48         movea.l 28(a1),a0          ; a0 := io_Data
5B4C         move.b  d1,(a0)+           ; CType in Datenbereich kopieren
5B4E         moveq   #1,d0              ; d0 := 1
5B50         move.l  d0,20(a1)          ; io_Actual := d0
5B54         moveq   #0,d0              ; d0 := ok-Kode
5B56         bra     FE5226             ; ---> IORequest beantworten
5B5A
5B5A ;-------------------------------   gpd_SetCType
5B5A
5B5A FE5B5A  move.l  24(a1),d0          ; d0 := io_Length
5B5E         subq.w  #1,d0              ; - 1
5B60         bcs     FE5CC2             ; < 0: Fehler ->
5B64         movem.l d2/a2,-(a7)        ; Register retten
5B68         movea.l 18(a1),a2          ; a2 := io_Unit
5B6C         move.l  a6,-(a7)           ; a6 retten
5B6E         movea.l 26(a6),a6          ; a6 := SysBase
5B72         jsr     -78(a6)            ; ---> Disable
5B76         movea.l (a7)+,a6           ; a6 wiederherstellen
5B78         lea     6E(a2),a0          ; a0 -> Input-Variablen
5B7C         moveq   #3,d0              ; d0 := 3 = Schleifenzähler
```

```
5B7E FE5B7E  clr.w   (aØ)+              ; Variable löschen
5B8Ø         dbra    dØ,FE5B7E          ; 3 mal wiederholen ->
5B84         movea.l 28(a1),aØ          ; aØ := io_Data
5B88         move.b  (aØ),dØ            ; dØ := CType
5B8A         ext.w   dØ                 ; auf Wort erweitern
5B8C         move.b  dØ,7B(a2)          ; und in gameport.device speichern
5B9Ø         clr.w   76(a2)             ; Eingabewerte bei Triggerung und
5B94         clr.w   78(a2)             ; Mauskoordinaten löschen
5B98         move.b  7A(a2),d1          ; d1 := Flags
5B9C         andi.w  #4,d1              ; PortBit isolieren
5BAØ         ori.b   #A,7A(a2)          ; Rel-Bit & SetCType-Flag setzen
5BA6         lsl.w   #2,dØ              ; dØ := Offset in Adressentabelle
5BA8         movea.l 8(pc,dØ.w),aØ      ; aØ := Adresse aus Tabelle FE5BB2
5BAC         jmp     (aØ)               ; ---> Routine aufrufen
5BAE
5BAE         DC.L    FE5BC2             ; -1  gpct_Allocated
5BB2 FE5BB2  DC.L    FE5BC2             ;  Ø  gpct_NoController
5BB6         DC.L    FE5CØE             ;  1  gpct_Mouse
5BBA         DC.L    FE5C1E             ;  2  gpct_RelJoyStick
5BBE         DC.L    FE5C3Ø             ;  3  gpct_AbsJoyStick
5BC2
5BC2 ;------ CType = Allocated oder NoController
5BC2
5BC2 FE5BC2  bsr.s   FE5BE8             ; ---> PotBits freigeben
5BC4         bsr.s   FE5BC8             ; ---> PotBits zuordnen
5BC6         bra.s   FE5C46             ; ---> Abschluß
5BC8
5BC8 ;------ PotBits zuordnen
5BC8
5BC8 FE5BC8  move.w  dØ,146(a6)         ;
5BCC         btst.l  #4,dØ              ; Startbit für rechten Port gesetzt?
5BDØ         beq.s   FE5BD6             ; nein: ->
5BD2         bset.l  #Ø,dØ              ; Startbit setzen
5BD6 FE5BD6  move.l  a6,-(a7)           ; a6 retten
5BD8         movea.l 34(a6),a6          ; a6 := PotgoBase
5BDC         jsr     -6(a6)             ; ---> AllocPotBits
5BEØ         movea.l (a7)+,a6           ; a6 wiederherstellen
5BE2         move.w  dØ,148(a6)         ; PotBit-Maske speichern
5BE6         rts
5BE8
```

```
5BE8 FE5BE8  move.w  148(a6),dØ          ; dØ := zugeordnete PotBits
5BEC         move.w  d1,d2               ; d1 in d2 retten
5BEE         move.l  a6,-(a7)            ; a6 retten
5BFØ         movea.l 34(a6),a6           ; a6 := PotgoBase
5BF4         jsr     -C(a6)              ; ---> FreePotBits
5BF8         movea.l (a7)+,a6            ; a6 wiederherstellen
5BFA         move.w  d2,d1               ; d1 wiederherstellen
5BFC         move.w  #FØ1,dØ             ; dØ := $FØ1 = PotBits und Startbit
5CØØ         lsl.w   d1,dØ               ; dØ := $FØ1 (Port 1) / $FØ1Ø (Port 2)
5CØ2         not.w   dØ                  ; dØ invertieren
5CØ4         and.w   dØ,14A(a6)          ; PotBits und Startbit löschen
5CØ8         and.w   146(a6),dØ          ;
5CØC         rts
5CØE
5CØE ;------ CType = Maus
5CØE
5CØE FE5CØE  movea.l #DFFØØØ,aØ         ; aØ := Basisadresse der Chipregister
5C14         lsr.w   #1,d1               ; d1 := Ø (Port 1) oder 2 (Port 2)
5C16         move.w  A(aØ,d1.w),78(a2)   ; Mausdaten aus JOYXDAT übernehmen
5C1C         add.w   d1,d1               ; d1 := Ø (Port 1) oder 4 (Port 2)
5C1E
5C1E ;------ CType = RelJoyStick
5C1E
5C1E FE5C1E  bsr.s   FE5BE8             ; ---> PotBits freigeben
5C2Ø         move.w  #FØØ,d2             ;
5C24         lsl.w   d1,d2               ; d2 := $FØØ (Port Ø) / $FØØØ (Port 1)
5C26         or.w    d2,dØ               ; in dØ einodern
5C28         or.w    d2,14A(a6)          ; in aktuelle PotBits einodern
5C2C         bsr.s   FE5BC8              ; ---> PotBits zuordnen
5C2E         bra.s   FE5C38              ; --->
5C3Ø
5C3Ø ;------ CType = AbsJoyStick
5C3Ø
5C3Ø FE5C3Ø  bclr.b  #3,7A(a2)           ; Rel-Flagbit löschen
5C36         bra.s   FE5C1E              ; --->
5C38
5C38 ;------ Abschluß
5C38
5C38 FE5C38  and.w   d2,dØ               ; CType-Bits in dØ entfernen
5C3A         cmp.w   d2,dØ               ; wurde dØ verändert?
```

```
5C3C          bne.s   FE5C40          ; ja: Fehler ->
5C3E          bra.s   FE5C46          ; --->
5C40
5C40 FE5C40   move.b  #1,1F(a1)       ; io_Error := 1 (invalid controller)
5C46 FE5C46   move.l  a6,-(a7)        ; a6 retten
5C48          movea.l 26(a6),a6       ; a6 := SysBase
5C4C          jsr     -7E(a6)         ; ---> Enable
5C50          movea.l (a7)+,a6        ; a6 wiederherstellen
5C52          moveq   #1,d0           ; d0 := 1
5C54          move.l  d0,20(a1)       ; io_Actual := d0
5C58          movem.l (a7)+,d2/a2     ; Register wiederherstellen
5C5C          moveq   #0,d0           ; d0 := ok-Kode
5C5E          bra     FE5226          ; ---> IORequest beantworten
5C62
5C62 ;--------------------------------  gpd_AskTrigger
5C62
5C62 FE5C62   move.l  24(a1),d0       ; d0 := io_Length
5C66          subq.w  #8,d0           ; - 8
5C68          bcs.s   FE5CC2          ; < 0: Fehler ->
5C6A          movem.l a2,-(a7)        ; a2 retten
5C6E          movea.l 18(a1),a2       ; a2 := io_Unit
5C72          lea     66(a2),a2       ; a2 -> Gameport Trigger
5C76          movea.l 28(a1),a0       ; a0 := io_Data
5C7A          moveq   #3,d0           ; d0 := 3 = Zähler für 4 Worte
5C7C FE5C7C   move.w  (a2)+,(a0)+     ; Triggerwort kopieren
5C7E          dbra    d0,FE5C7C       ; 3 mal wiederholen ->
5C82          movem.l (a7)+,a2        ; a2 wiederherstellen
5C86          moveq   #8,d0           ; d0 := 8
5C88          move.l  d0,20(a1)       ; io_Actual := d0
5C8C          moveq   #0,d0           ; d0 := 0 = ok-Flag
5C8E          bra     FE5226          ; ---> IORequest beantworten
5C92
5C92 ;--------------------------------  gpd_SetTrigger
5C92
5C92 FE5C92   move.l  24(a1),d0       ; d0 := io_Length
5C96          subq.w  #8,d0           ; - 8
5C98          bcs.s   FE5CC2          ; < 0: Fehler ->
5C9A          movem.l a2,-(a7)        ; a2 retten
5C9E          movea.l 18(a1),a2       ; a2 := io_Unit
5CA2          lea     66(a2),a2       ; a2 -> Gameport Trigger
```

```
5CA6          movea.l 28(a1),aØ          ; aØ := io_Data
5CAA          moveq   #3,dØ              ; dØ := 3 = Zähler für 4 Worte
5CAC FE5CAC   move.w  (aØ)+,(a2)+        ; Triggerwort kopieren
5CAE          dbra    dØ,FE5CAC          ; 3 mal wiederholen ->
5CB2          movem.l (a7)+,a2           ; a2 wiederherstellen
5CB6          moveq   #8,dØ              ; dØ := 8
5CB8          move.l  dØ,2Ø(a1)          ; io_Actual := dØ
5CBC          moveq   #Ø,dØ              ; dØ := Ø = ok-Flag
5CBE          bra     FE5226             ; ---> IORequest beantworten
5CC2
5CC2 FE5CC2   moveq   #-4,dØ             ; dØ := ioerr_BadLength
5CC4          bra     FE5226             ; ---> IORequest beantworten
5CC8
5CC8 ;--------------------------------   gpd_InputHandler
5CC8
5CC8 FE5CC8   movea.l a1,a6              ; a6 := a1 -> gameport.device
5CCA          lea     38(a6),aØ          ; aØ -> Unit 1 (linker Port)
5CCE          bsr.s   FE5CF8             ; ---> PortServer
5CDØ          lea     B4(a6),aØ          ; aØ -> Unit 2 (rechter Port)
5CD4          bsr.s   FE5CF8             ; ---> PortServer
5CD6          move.w  14A(a6),dØ         ; dØ := PotBits
5CDA          btst.l  #4,dØ              ; Startbit für rechten Port gesetzt?
5CDE          beq.s   FE5CE4             ; nein: ->
5CEØ          bset.l  #Ø,dØ              ; Startbit setzen
5CE4 FE5CE4   move.w  148(a6),d1         ; d1 := zugeordnete PotBits
5CE8          move.l  a6,-(a7)           ; a6 retten
5CEA          movea.l 34(a6),a6          ; a6 := PotgoBase
5CEE          jsr     -12(a6)            ; ---> WritePotgo
5CF2          movea.l (a7)+,a6           ; a6 wiederherstellen
5CF4          clr.w   dØ                 ; dØ := Ø
5CF6          rts
5CF8
5CF8 ;------ PortServer
5CF8
5CF8 FE5CF8   movem.l d2-d4/a2-a4,-(a7)  ; Register retten
5CFC          movea.l aØ,a2              ; a2 := aØ = io_Unit -> MsgPort
5CFE          bclr.b  #1,7A(a2)          ; SetCType-Flag rücksetzen
5DØ4          bne.s   FE5D3A             ; war gesetzt: SetCType aktiv ->
5DØ6          addq.w  #1,7Ø(a2)          ; Zeitzähler inkrementieren
5DØA          bne.s   FE5D1Ø             ; noch kein Überlauf: ->
```

```
5D0C          subq.w  #1,70(a2)           ; auf $FFFF zurücksetzen
5D10 FE5D10   move.b  7B(a2),d0           ; d0 := CType
5D14          ext.w   d0                  ; auf Wort erweitern
5D16          cmpi.w  #1,d0               ; Controller definiert?
5D1A          blt.s   FE5D3A              ; nein: Fertig ->
5D1C          cmpi.w  #3,d0               ; CType zulässig?
5D20          bgt.s   FE5D3A              ; nein: Fertig ->
5D22          move.b  7A(a2),d1           ; d1 := Flags
5D26          andi.w  #4,d1               ; PortBit isolieren
5D2A          lsl.w   #2,d0               ; d0 := CType * 4 = Tabellenoffset
5D2C          movea.l E(pc,d0.w),a1       ; a1 := FE5D40-4 (Offset >= 4!)
5D30          movea.l #DFF000,a0          ; a0 := Basisadresse der Chipregister
5D36          jsr     (a1)                ; ---> Input-Routine aufrufen
5D38          bra.s   FE5D4C              ; ---> Triggerung
5D3A
5D3A ;------ Abschluß
5D3A
5D3A FE5D3A   movem.l (a7)+,d2-d4/a2-a4   ; Register wiederherstellen
5D3E          rts
5D40
5D40 ;------ Adressentabelle der Input-Routinen
5D40
5D40 FE5D40   DC.L    FE5EF4              ; Maus
5D44          DC.L    FE5F3A              ; RelJoyStick
5D48          DC.L    FE5F3A              ; AbsJoyStick
5D4C
5D4C ;------ Triggerung
5D4C
5D4C FE5D4C   moveq   #0,d3               ; d3 := 0
5D4E          btst.b  #3,7A(a2)           ; Rel-Flag gesetzt?
5D54          bne.s   FE5D9E              ; ja: ->
5D56          move.b  d0,d1               ; d1 := d0.b = X-Input
5D58          ext.w   d1                  ; auf Wort erweitern
5D5A          move.w  d1,72(a2)           ; und speichern
5D5E          sub.b   77(a2),d1           ; letzten Triggerwert abziehen
5D62          bpl.s   FE5D66              ; Ergebnis nicht negativ: ->
5D64          neg.b   d1                  ; Ergebnis positiv machen
5D66 FE5D66   move.w  6A(a2),d2           ; d2 := gpt_XDelta
5D6A          beq.s   FE5D78              ; = 0: Keine Triggerung ->
5D6C          ext.w   d1                  ; auf Wort erweitern (evtl. negiert!)
```

```
5D6E          cmp.w   d1,d2           ; Triggerschwelle > Input?
5D7Ø          bgt.s   FE5D78          ; ja: Keine Triggerung ->
5D72          move.b  dØ,77(a2)       ; Triggerwert abspeichern
5D76          st      d3              ; Triggerflag d3 setzen
5D78 FE5D78   lsr.w   #8,dØ           ; dØ.b := Y-Input
5D7A          move.b  dØ,d1           ; d1 := dØ
5D7C          ext.w   d1              ; auf Wort erweitern
5D7E          move.w  d1,74(a2)       ; und speichern
5D82          sub.b   76(a2),d1       ; letzten Triggerwert abziehen
5D86          bpl.s   FE5D8A          ; Ergebnis nicht negativ: ->
5D88          neg.b   d1              ; Ergebnis positiv machen
5D8A FE5D8A   move.w  6C(a2),d2       ; d2 := gpt_YDelta
5D8E          beq.s   FE5DD4          ; = Ø: Keine Triggerung ->
5D9Ø          ext.w   d1              ; auf Wort erweitern (evtl. negiert!)
5D92          cmp.w   d1,d2           ; Triggerschwelle > Input?
5D94          bgt.s   FE5DD4          ; ja: Keine Triggerung ->
5D96          move.b  dØ,76(a2)       ; Triggerwert abspeichern
5D9A          st      d3              ; Triggerflag d3 setzen
5D9C          bra.s   FE5DD4          ; ---> Maustastenstatus bearbeiten
5D9E
5D9E FE5D9E   move.b  dØ,d1           ; d1 := dØ.b = X-Input
5DAØ          ext.w   d1              ; auf Wort erweitern
5DA2          add.w   72(a2),d1       ; letzten X-Wert addieren
5DA6          move.w  d1,72(a2)       ; neuen X-Wert abspeichern
5DAA          bpl.s   FE5DAE          ; Ergebnis nicht negativ: ->
5DAC          neg.w   d1              ; Ergebnis positiv machen
5DAE FE5DAE   move.w  6A(a2),d2       ; d2 := gpt_XDelta
5DB2          beq.s   FE5DB8          ; = Ø: Keine Triggerung ->
5DB4          cmp.w   d1,d2           ; Triggerschwelle > X-Wert?
5DB6          sle     d3              ; nein: Triggerflag setzen
5DB8 FE5DB8   lsr.w   #8,dØ           ; dØ.b := Y-Input
5DBA          ext.w   dØ              ; auf Wort erweitern
5DBC          add.w   74(a2),dØ       ; letzten Y-Wert addieren
5DCØ          move.w  dØ,74(a2)       ; neuen Y-Wert abspeichern
5DC4          bpl.s   FE5DC8          ; Ergebnis nicht negativ: ->
5DC6          neg.w   dØ              ; Ergebnis positiv machen
5DC8 FE5DC8   move.w  6C(a2),d2       ; d2 := gpt_YDelta
5DCC          beq.s   FE5DD4          ; = Ø: Keine Triggerung ->
5DCE          cmp.w   dØ,d2           ; Triggerschwelle > Y-Wert?
5DDØ          bgt.s   FE5DD4          ; ja: Keine Triggerung ->
```

```
5DD2          st      d3                  ; Triggerflag setzen
5DD4
5DD4 ;------ Maustastenstatus bearbeiten
5DD4
5DD4 FE5DD4   move.w  24(a2),d1           ; d1 := BufferHead
5DD8          addq.w  #8,d1               ; um 8 erhöhen
5DDA          andi.w  #3F,d1              ; mod Pufferlänge 64
5DDE          cmp.w   22(a2),d1           ; Puffer voll?
5DE2          beq     FE5D3A              ; ja: Fertig ->
5DE6          swap    d0                  ; d0.w := Maustastenflags (Bit 13,14,15)
5DE8          moveq   #FF,d4              ; d4 := -1 = iecode_NoButton
5DEA          move.w  d0,d1               ; d1 := d0 = Maustastenflags
5DEC          move.w  6E(a2),d2           ; d2 := letzter Maustastenzustand
5DF0          eor.w   d2,d1               ; = neuer Zustand?
5DF2          beq     FE5E6A              ; ja: Keine Aktion erforderlich ->
5DF6          and.w   d0,d1               ; wurde Taste gedrückt?
5DF8          beq.s   FE5E30              ; nein: ->
5DFA          btst.b  #0,67(a2)           ; gpt_Keys: Downkeys gesetzt?
5E00          beq.s   FE5E2C              ; nein: Keine Triggerung ->
5E02          moveq   #F,d2               ; d2 := 15 = Zähler für Schiebeschleife
5E04 FE5E04   lsl.w   #1,d1               ; d1 um 1 Bit nach links verschieben
5E06          dbcs    d2,FE5E04           ; bis CF gesetzt oder d2 = -1: ->
5E0A          move.b  10(pc,d2.w),d4      ; d4 := Kode aus Tabelle FE5E1C
5E0E          moveq   #FF,d3              ; Triggerflag setzen
5E10          move.w  6E(a2),d1           ; d1 := Letzter Mauszustand
5E14          bset.l  d2,d1               ; Neues Maustastenflag setzen
5E16          move.w  d1,6E(a2)           ; und wieder abspeichern
5E1A          bra.s   FE5E7E              ; --->
5E1C
5E1C FE5E1C   DC.B    50 51 52 53 54 55 56 57 58 59 5A 5B FF 6A 69 68
5E2C
5E2C FE5E2C   or.w    d1,6E(a2)           ; Flags für gedrückte Maustasten setzen
5E30 FE5E30   move.w  d0,d1               ; d1 := d0 = aktueller Maustastenstatus
5E32          move.w  6E(a2),d2           ; d2 := alter Maustastenstatus
5E36          eor.w   d2,d1               ; d1 := alle geänderten Flags
5E38          not.w   d0                  ; d0 invertieren
5E3A          and.w   d0,d1               ; d1 := von 1 nach 0 geänderte Flags
5E3C          beq.s   FE5E6A              ; Kein Flag: Keine Triggerung ->
5E3E          btst.b  #1,67(a2)           ; gpt_Keys: Upkeys gesetzt?
5E44          beq.s   FE5E64              ; nein: Keine Triggerung ->
```

```
5E46          moveq   #F,d2               ; d2 := 15 = Zähler für Schiebeschleife
5E48 FE5E48   lsl.w   #1,d1               ; d1 um 1 Bit nach links verschieben
5E4A          dbcs    d2,FE5E48           ; bis CF gesetzt oder d2 = -1: ->
5E4E          move.b  -34(pc,d2.w),d4     ; d4 := Kode aus Tabelle FE5E1C
5E52          ori.b   #80,d4              ; Bit 7 (Upkey-Flag) setzen
5E56          moveq   #FF,d3              ; Triggerflag setzen
5E58          move.w  6E(a2),d1           ; d1 := Maustastenstatus
5E5C          bclr.l  d2,d1               ; Tastenflag löschen
5E5E          move.w  d1,6E(a2)           ; Status wieder speichern
5E62          bra.s   FE5E6A              ; ---> Triggerung auslösen
5E64
5E64 FE5E64   not.w   d1                  ; d1 invertieren
5E66          and.w   d1,6E(a2)           ; Neuen Maustastenstatus speichern
5E6A FE5E6A   tst.b   d3                  ; Triggerflag gesetzt?
5E6C          bne.s   FE5E7E              ; ja: ->
5E6E          move.w  68(a2),d0           ; d0 := gpt_Timeout
5E72          beq     FE5D3A              ; = 0: Keine Triggerung ->
5E76          cmp.w   70(a2),d0           ; Zeitzähler < Triggerschwelle?
5E7A          bhi     FE5D3A              ; ja: Keine Triggerung ->
5E7E FE5E7E   move.l  a6,-(a7)            ; a6 retten
5E80          movea.l 26(a6),a6           ; a6 := SysBase
5E84          jsr     -78(a6)             ; ---> Disable
5E88          movea.l (a7)+,a6            ; a6 wiederherstellen
5E8A          move.w  24(a2),d1           ; d1 := BufferHead
5E8E          move.b  d4,26(a2,d1.w)      ; Maustastenkode in Puffer schreiben
5E92          move.b  6E(a2),d0           ; d0.b := Maustastenstatus
5E96          lsr.b   #1,d0               ; um 1 Bit nach rechts schieben
5E98          andi.b  #70,d0              ; Bit 7 löschen
5E9C          btst.b  #3,7A(a2)           ; Rel-Flag testen
5EA2          beq.s   FE5EA8              ; nicht gesetzt: ->
5EA4          ori.b   #80,d0              ; Bit 7 setzen
5EA8 FE5EA8   move.b  d0,27(a2,d1.w)      ; Ergebnis in Puffer schreiben
5EAC          move.w  72(a2),28(a2,d1.w)  ; X-Wert in Puffer
5EB2          move.w  74(a2),2A(a2,d1.w)  ; Y-Wert in Puffer
5EB8          move.w  70(a2),2C(a2,d1.w)  ; Zeitzählerstand in Puffer
5EBE          addq.w  #8,d1               ; Pufferoffset um 8 erhöhen
5EC0          andi.w  #3F,d1              ; mod 64 = Pufferlänge
5EC4          move.w  d1,24(a2)           ; als neuen BufferHead speichern
5EC8          clr.w   70(a2)              ; Zeitzähler löschen
5ECC          clr.w   72(a2)              ; X-Wert löschen
```

```
5ED0          clr.w   74(a2)              ; Y-Wert löschen
5ED4          move.l  a6,-(a7)            ; a6 retten
5ED6          movea.l 26(a6),a6           ; a6 := SysBase
5EDA          jsr     -7E(a6)             ; ---> Enable
5EDE          movea.l (a7)+,a6            ; a6 wiederherstellen
5EE0          movea.l 14(a2),a1           ; a1 := MsgList.lh_Head
5EE4          tst.l   (a1)                ; Ende der Liste?
5EE6          beq.s   FE5EF0              ; ja: Fertig ->
5EE8          bsr     FE5A2A              ; ---> gpd_ReadEvent
5EEC          bra     FE5D3A              ; ---> Abschluß
5EF0
5EF0 FE5EF0   bra     FE5D3A              ; ---> Abschluß
5EF4
5EF4 ;------ Input-Routine für Maus
5EF4
5EF4 FE5EF4   bsr     FE5F24              ; ---> POTINP lesen und aufbereiten
5EF8          lsr.w   #1,d1               ; d1 := 0 (Port 1) oder 2 (Port 2)
5EFA          move.w  A(a0,d1.w),d0       ; d0 := JOYXDAT = Mauszählerwert
5EFE          move.w  d0,d2               ; d2 := d0 = Mauszählerwert
5F00          ror.w   #8,d0               ; d0.b := Maus_Y
5F02          sub.b   78(a2),d0           ; - letztes Maus_Y
5F06          ror.w   #8,d0               ; d0.b := Maus_X
5F08          sub.b   79(a2),d0           ; - letztes Maus_X
5F0C          move.w  d2,78(a2)           ; Neue Mauszählerwerte speichern
5F10          add.w   d1,d1               ; d1 := 0 (Port 1) oder 4 (Port 2)
5F12 FE5F12   lsr.w   #2,d1               ; d1 := 0 (Port 1) oder 1 (Port 2)
5F14          addq.w  #6,d1               ; d1 := Feuerknopf-Bit für Port
5F16          btst.b  d1,BFE001           ; Feuerknopf/Linke Maustaste gedrückt?
5F1C          bne.s   FE5F22              ; nein: ->
5F1E          bset.l  #1F,d0              ; Bit 31 von d0 setzen
5F22 FE5F22   rts
5F24
5F24 ;------ POTINP lesen und aufbereiten
5F24
5F24 FE5F24   move.w  16(a0),d0           ; d0 := POTINP
5F28          lsr.w   d1,d0               ; 4-Bit-Rechtsverschiebung für Port 2
5F2A          not.w   d0                  ; Bits flippen
5F2C          andi.w  #500,d0             ; Input-Bits:    00000Y0X 00000000
5F30          rol.w   #7,d0               ; Wort rotieren: X0000000 0000000Y0
5F32          ror.b   #1,d0               ; Byte rotieren: X0000000 0000000Y
```

```
5F34          ror.w   #2,d0                ; Wort rotieren: 0YX00000 00000000
5F36          swap    d0                   ; und im oberen Wort von d0 ablegen
5F38          rts
5F3A
5F3A ;------ Input-Routine für Joystick
5F3A
5F3A FE5F3A   bsr.s   FE5F24               ; ---> POTINP lesen und aufbereiten
5F3C          move.w  d1,d0                ; d0 := d1 = 0 (Port 1) oder 4 (Port 2)
5F3E          lsr.w   #1,d0                ; d0 := 0 (Port 1) oder 2 (Port 2)
5F40          move.w  A(a0,d0.w),d0        ; d0 := JOYXDAT
5F44          andi.w  #303,d0              ; Input-Bits:    000000YY000000XX
5F48          ror.b   #2,d0                ; Byte rotieren: 000000YYXX000000
5F4A          lsr.w   #5,d0                ; Wort rotieren: 0000000000YYXX0
5F4C          move.w  4(pc,d0.w),d0        ; d0 := Wort aus Tabelle FE5F52
5F50          bra.s   FE5F12               ; ---> Feuerknopf übernehmen
5F52
5F52 ;------ Tabelle der Joystick-Inputs
5F52 ;
5F52 ; Byte 1: -1 = Vorn, 0 = Mitte, 1 = Hinten
5F52 ; Byte 2: -1 = Links, 0 = Mitte, 1 = Rechts
5F52
5F52 FE5F52   DC.B    0,0          ; YYXX = 0000: Mitte
5F54          DC.B    1,0          ;        0001: Hinten
5F56          DC.B    1,1          ;        0010: Hinten & Rechts
5F58          DC.B    0,1          ;        0011: Rechts
5F5A          DC.B    -1,0         ;        0100: Vorn
5F5C          DC.B    0,0          ;        0101: V & H = Mitte
5F5E          DC.B    0,1          ;        0110: V & H & R = Rechts
5F60          DC.B    -1,1         ;        0111: Vorn & Rechts
5F62          DC.B    -1,-1        ;        1000: Vorn & Links
5F64          DC.B    0,0          ;        1001: V & L & H = Mitte?? (Egal)
5F66          DC.B    0,0          ;        1010: V & L & H & R = Mitte
5F68          DC.B    -1,0         ;        1011: V & L & R = Vorn
5F6A          DC.B    0,-1         ;        1100: Links
5F6C          DC.B    1,-1         ;        1101: Links & Hinten
5F6E          DC.B    1,0          ;        1110: L & H & R = Hinten
5F70          DC.B    0,0          ;        1111: L & R = Mitte
5F72          DC.W    0
5F74
5F74 ************************************************************************
```

Input.device Datenbereich

```
-Ø24 ************************** Input.device ******************************
-Ø24
-Ø24          jmp    FE5220       ; ---> AbortIO
-Ø1E          jmp    FE515C       ; ---> BeginIO
-Ø18          jmp    FE52F4       ; ---> Extfunct
-Ø12          jmp    FE61AE       ; ---> Expunge
-ØØC          jmp    FE52F6       ; ---> Close
-Ø06          jmp    FE61BA       ; ---> Open
ØØØØ
ØØØØ ;-------------------------   Library-Block
ØØØØ
ØØØØ indBase DC.L    45B8         ; ln_Succ -> console.device  \  lib_Node
ØØØ4          DC.L    2FDC         ; ln_Pred -> audio.device     |
ØØØ8          DC.B    3            ; ln_Type (3 = Device)        |
ØØØ9          DC.B    3C           ; ln_Pri                      |
ØØØA          DC.L    FE5ØE0       ; ln_Name -> 'input.device'  /
ØØØE
ØØØE          DC.B    6            ; lib_Flags (Changed, Sumused)
ØØØF          DC.B    Ø            ; lib_Pad
ØØ1Ø          DC.W    24           ; lib_NegSize = 36 Bytes
ØØ12          DC.W    1232         ; lib_PosSize = 4658 Bytes
ØØ14          DC.W    21           ; lib_Version = 33
ØØ16          DC.W    67           ; lib_Revision = 1Ø3
ØØ18          DC.L    Ø            ; lib_IdString
ØØ1C          DC.L    EC98ØØØØ     ; lib_LibSum
ØØ2Ø          DC.W    4            ; lib_OpenCnt
ØØ22
ØØ22 ;-------------------------   Verschiedene Parameter
ØØ22
ØØ22          DC.L    Ø            ; SegList
ØØ26          DC.L    676          ; SysBase
ØØ2A          DC.L    FE6142       ; CmdArray (Tabelle der IO-Kommandos)
```

```
002E          DC.L    FE6186        ; CmdFlags (Tabelle der MsgPort-Nummern)
0032          DC.W    11            ; LastCmd (Höchste Kommandonummer + 1)
0034
0034 ;-------------------------   MessagePort für Kommandos (CommandPort)
0034
0034    00    DC.L    0             ; ln_Succ                     \  mp_Node
0038    04    DC.L    0             ; ln_Pred                     |
003C    08    DC.B    4             ; ln_Type (4 = MsgPort)       |
003D    09    DC.B    0             ; ln_Pri (als Flag verwendet) |
003E    0A    DC.L    FE50E0        ; ln_Name -> 'input.device'   /
0042
0042    0E    DC.B    0             ; mp_Flags
0043    0F    DC.B    1F            ; mp_SigBit
0044    10    DC.L    *0056         ; mp_SigTask -> input task
0048
0048    14    DC.L    *004C         ; lh_Head      \  mp_MsgList
004C    18    DC.L    0             ; lh_Tail      |
0050    1C    DC.L    *0048         ; lh_TailPred  |
0054    20    DC.B    0             ; lh_Type      |
0055    21    DC.B    0             ; lh_Pad       /
0056
0056 ;-------------------------   InputTask
0056
0056    00    DC.L    081E          ; ln_Succ                   \  tc_Node
005A    04    DC.L    6166          ; ln_Pred                   |
005E    08    DC.B    1             ; ln_Type (1 = Task)        |
005F    09    DC.B    14            ; ln_Pri                    |
0060    0A    DC.L    FE50E0        ; ln_Name -> input.device   /
0064
0064    0E    DC.B    0             ; tc_Flags
0065    0F    DC.B    4             ; tc_State (4 = ts_Wait)
0066    10    DC.B    0             ; tc_IDNestCnt
0067    11    DC.B    -1            ; tc_TDNestCnt
0068    12    DC.L    C000FFFF      ; tc_SigAlloc
006C    16    DC.L    C0000000      ; tc_SigWait
0070    1A    DC.L    0             ; tc_SigRecvd
0074    1E    DC.L    0             ; tc_SigExcept
0078    22    DC.W    8000          ; tc_TrapAlloc
007A    24    DC.W    0             ; tc_TrapAble
007C    26    DC.L    0             ; tc_ExceptData
```

```
0080    2A    DC.L    FC2FB4        ; tc_ExceptCode
0084    2E    DC.L    0             ; tc_TrapData
0088    32    DC.L    FC2FB4        ; tc_TrapCode
008C    36    DC.L    *105C         ; tc_SPReg
0090    3A    DC.L    *00B2         ; tc_SPLower
0094    3E    DC.L    *10B2         ; tc_SPUpper
0098    42    DC.L    0             ; tc_Switch
009C    46    DC.L    0             ; tc_Launch
00A0    4A    DC.L    0             ; lh_Head        \  tc_MemEntry
00A4    4E    DC.L    0             ; lh_Tail        |
00A8    52    DC.L    0             ; lh_TailPred    |
00AC    56    DC.B    0             ; lh_Type        |
00AD    57    DC.B    0             ; lh_Pad         /
00AE    58    DC.L    0             ; tc_UserData
00B2
00B2          DS.W    800           ; TaskStack (2048 Worte)
10B2
10B2 ;-------------------------   InputHandler-Liste
10B2
10B2    00    DC.L    4AC6          ; lh_Head
10B6    04    DC.L    0             ; lh_Tail
10BA    08    DC.L    463A          ; lh_TailPred
10BE    0C    DC.B    0             ; lh_Type
10BF    0D    DC.B    0             ; lh_Pad
10C0
10C0 ;-------------------------   ReplyPort
10C0
10C0    00    DC.L    0             ; ln_Succ                \  mp_Node
10C4    04    DC.L    0             ; ln_Pred                |
10C8    08    DC.B    4             ; ln_Type (4=nt_MsgPort) |
10C9    09    DC.B    0             ; ln_Pri                 |
10CA    0A    DC.L    0             ; ln_Name                /
10CE    0E    DC.B    0             ; mp_Flags
10CF    0F    DC.B    1E            ; mp_SigBit
10D0    10    DC.L    *0056         ; mp_SigTask -> InputTask
10D4    14    DC.L    *10D8         ; lh_Head                \  mp_MsgList
10D8    18    DC.L    0             ; lh_Tail                |
10DC    1C    DC.L    *10D4         ; lh_TailPred            |
10E0    20    DC.B    0             ; lh_Type                |
10E1    21    DC.B    0             ; lh_Pad                 /
```

```
10E2
10E2 ;-------------------------   TimerEvent
10E2
10E2    00   DC.L   0            ; ie_NextEvent
10E6    04   DC.B   6            ; ie_Class = ieClass_Timer
10E7    05   DC.B   0            ; ie_SubClass = 0
10E8    06   DC.W   0            ; ie_Code = 0
10EA    08   DC.W   0            ; ie_Qualifier
10EC    0A   DC.L   0            ; ie_EventAddress
10F0    0E   DC.L   11C65B95     ; ie_TimeStamp.tv_Secs
10F4    12   DC.L   000D764C     ; ie_TimeStamp.tv_Micro
10F8
10F8 ;-------------------------   TimerEvent-Request
10F8
10F8    00   DC.L   1E408        ; ln_Succ
10FC    04   DC.L   2F56         ; ln_Pred
1100    08   DC.B   5            ; ln_Type (5=nt_Message)
1101    09   DC.B   0            ; ln_Pri
1102    0A   DC.L   0            ; ln_Name
1106    0E   DC.L   *10C0        ; mn_ReplyPort
110A    12   DC.W   0            ; mn_Length
110C    14   DC.L   2EDE         ; io_Device -> timer.device
1110    18   DC.L   2F56         ; io_Unit -> Unit_VBlank
1114    1C   DC.W   9            ; io_Command = tr_AddRequest
1116    1E   DC.B   0            ; io_Flags
1117    1F   DC.B   0            ; io_Error
1118    20   DC.L   0            ; tr_Secs
111C    24   DC.L   0186A0       ; tr_Micro = 100000
1120
1120 ;-------------------------   Gameport-Event
1120
1120    00   DC.L   0            ; ie_NextEvent
1124    04   DC.B   2            ; ie_Class = ieclass_RawMouse
1125    05   DC.B   0            ; ie_SubClass
1126    06   DC.W   00FF         ; ie_Code
1128    08   DC.W   8000         ; ie_Qualifier
112A    0A   DC.W   0            ; ie_X
112C    0C   DC.W   -1           ; ie_Y
112E    0E   DC.L   11C65C1A     ; ie_TimeStamp.tv_Secs
1132    12   DC.L   000369EF     ; ie_TimeStamp.tv_Micro
```

```
1136
1136 ;-------------------------   Gameport-ReadRequest
1136
1136    ØØ   DC.L   2DA4          ; ln_Succ
113A    Ø4   DC.L   2DAØ          ; ln_Pred
113E    Ø8   DC.B   5             ; ln_Type (5=nt_Message)
113F    Ø9   DC.B   Ø             ; ln_Pri
114Ø    ØA   DC.L   Ø             ; ln_Name
1144    ØE   DC.L   *1ØCØ         ; mn_ReplyPort
1148    12   DC.W   Ø             ; mn_Length
114A    14   DC.L   2D54          ; io_Device -> gameport.device
114E    18   DC.L   2D8C          ; io_Unit -> Gameport Ø
1152    1C   DC.W   9             ; io_Command
1154    1E   DC.B   2Ø            ; io_Flags
1155    1F   DC.B   Ø             ; io_Error
1156    2Ø   DC.L   16            ; io_Actual
115A    24   DC.L   16            ; io_Length
115E    28   DC.L   *112Ø         ; io_Data
1162    2C   DC.L   Ø             ; io_Offset
1166
1166 ;-------------------------   Keyboard-Event
1166
1166    ØØ   DC.L   Ø             ; ie_NextEvent
116A    Ø4   DC.B   1             ; ie_Class
116B    Ø5   DC.B   Ø             ; ie_SubClass
116C    Ø6   DC.W   C4            ; ie_Code
116E    Ø8   DC.W   8ØØØ          ; ie_Qualifier
117Ø    ØA   DC.L   Ø             ; ie_EventAddress
1174    ØE   DC.L   11C65F49      ; ie_TimeStamp.tv_Secs
1178    12   DC.L   ØØØ9ØF Ø4     ; ie_TimeStamp.tv_Micro
117C
117C ;-------------------------   Keyboard-ReadRequest
117C
117C    ØØ   DC.L   2C68          ; ln_Succ
118Ø    Ø4   DC.L   2C64          ; ln_Pred
1184    Ø8   DC.B   5             ; ln_Type (5=nt_Message)
1185    Ø9   DC.B   Ø             ; ln_Pri
1186    ØA   DC.L   Ø             ; ln_Name
```

```
118A    ØE   DC.L   *1ØCØ        ; mn_ReplyPort
118E    12   DC.W   Ø             ; mn_Length
1190    14   DC.L   2C1C          ; io_Device -> keyboard.device
1194    18   DC.L   2C5Ø          ; io_Unit -> MsgPort im keyboard.device
1198    1C   DC.W   9             ; io_Command
119A    1E   DC.B   2Ø            ; io_Flags
119B    1F   DC.B   Ø             ; io_Error
119C    2Ø   DC.L   16            ; io_Actual
11AØ    24   DC.L   16            ; io_Length
11A4    28   DC.L   *1166         ; io_Data
11A8    2C   DC.L   Ø             ; io_Offset
11AC
11AC ;-------------------------   TimeRequest für Threshold/Period
11AC
11AC    ØØ   DC.L   4354          ; ln_Succ
11BØ    Ø4   DC.L   435Ø          ; ln_Pred
11B4    Ø8   DC.B   7             ; ln_Type (7=nt_ReplyMsg)
11B5    Ø9   DC.B   Ø             ; ln_Pri
11B6    ØA   DC.L   Ø             ; ln_Name
11BA    ØE   DC.L   *1ØCØ        ; mn_ReplyPort
11BE    12   DC.W   Ø             ; mn_Length
11CØ    14   DC.L   2EDE          ; io_Device -> timer.device
11C4    18   DC.L   2F56          ; io_Unit -> Unit_VBlank
11C8    1C   DC.W   9             ; io_Command = tr_AddRequest
11CA    1E   DC.B   Ø             ; io_Flags
11CB    1F   DC.B   FE            ; io_Error
11CC    2Ø   DC.L   Ø             ; tr_Secs
11DØ    24   DC.L   ØØØA122Ø      ; tr_Micro = 66ØØØØ
11D4
11D4 ;-------------------------   Timer GetSysTime-Request
11D4
11D4    ØØ   DC.L   Ø             ; ln_Succ
11D8    Ø4   DC.L   Ø             ; ln_Pred
11DC    Ø8   DC.B   5             ; ln_Type (5=nt_Message)
11DD    Ø9   DC.B   Ø             ; ln_Pri
11DE    ØA   DC.L   Ø             ; ln_Name
11E2    ØE   DC.L   *1ØCØ        ; mn_ReplyPort
11E6    12   DC.W   Ø             ; mn_Length
11E8    14   DC.L   2EDE          ; io_Device -> timer.device
11EC    18   DC.L   2F56          ; io_Unit -> Unit_VBlank
```

```
11F0    1C   DC.W   A            ; io_Command
11F2    1E   DC.B   1            ; io_Flags
11F3    1F   DC.B   0            ; io_Error
11F4    20   DC.L   11C660D2     ; tr_Secs
11F8    24   DC.L   0004DBF7     ; tr_Micro
11FC
11FC ;-------------------------   Parameter für Threshold/Period
11FC
11FC         DC.L   0            ; Threshold Sekunden
1200         DC.L   0C3500       ;           Mikrosekunden = 800000
1204
1204         DC.L   0            ; Period Sekunden
1208         DC.L   0186A0       ;        Mikrosekunden = 100000
120C
120C ;-------------------------   RepeatEvent für Tastenwiederholung
120C
120C    00   DC.L   0            ; ie_NextEvent
1210    04   DC.B   1            ; ie_Class = ieclass_RawKey
1211    05   DC.B   0            ; ie_SubClass
1212    06   DC.W   -1           ; ie_Code
1214    08   DC.W   8200         ; ie_Qualifier
1216    0A   DC.W   2800         ; ie_X
1218    0C   DC.W   4100         ; ie_Y
121A    0E   DC.L   11C6612B     ; ie_TimeStamp.tv_Secs
121E    12   DC.L   0006B4AC     ; ie_TimeStamp.tv_Micro
1222
1222 ;-------------------------   Letzter Qualifier und KeyStack für DeadKeys
1222
1222         DC.W   8000         ; Qualifier (Maus & Keyboard)
1224
1224         DC.B   44           ; Keycode
1225         DC.B   0            ; Qualifier (Low Byte)
1226         DC.B   44           ; letzter Keycode
1227         DC.B   0            ; letzter Qualifier (Low Byte)
1228
1228 ;-------------------------   Port, ControllerType und Triggerparameter
1228
1228         DC.B   0            ; MPort
1229         DC.B   1            ; MType
122A
```

```
122A          DC.W   3              ; MTrig  gpt_Keys
122C          DC.W   0              ;        gpt_Timeout
122E          DC.W   1              ;        gpt_XDelta
1230          DC.W   1              ;        gpt_YDelta
1232
1232 ************************************************************************
```

Input.device Routinen

```
5F74 *************** Input.device Routinen ********************************
5F74
5F74 ;---------------------------------   Initialisierung input.device
5F74
5F74 FE5F74  movem.l aØ/a2,-(a7)          ; Register retten
5F78         lea     1B8(pc),aØ           ; aØ := FE6132 -> Vektortabelle
5F7C         lea     11A(pc),a1           ; a1 := FE6Ø98 -> Tabelle f InitStruct
5F8Ø         lea     C(pc),a2             ; a2 := FE5F8E -> Init-Routine
5F84         move.l  #1232,dØ             ; dØ := 4658 Bytes Speicherbedarf
5F8A         bra     FE52A4               ; ---> Device einrichten
5F8E
5F8E ;------ Init-Routine für MakeLibrary
5F8E
5F8E FE5F8E  movem.l a2-a3/a6,-(a7)       ; Register retten
5F92         movea.l dØ,aØ                ; aØ := dØ -> Input.device
5F94         move.l  a6,26(aØ)            ; SysBase speichern
5F98         movea.l dØ,a6                ; a6 := dØ -> Input.device
5F9A         lea     48(a6),aØ            ; aØ -> MsgList im CommandPort
5F9E         move.l  aØ,(aØ)              ; Macro NewList
5FAØ         addq.l  #4,(aØ)              ; :
5FA2         clr.l   4(aØ)                ; :
5FA6         move.l  aØ,8(aØ)             ; :
5FAA         lea     CE(pc),aØ            ; aØ := FE6Ø7A -> 'timer.device'
5FAE         moveq   #1,dØ                ; dØ := 1 = UnitNumber (Unit_VBlank)
5FBØ         moveq   #Ø,d1                ; d1 := Ø = Flags
5FB2         lea     11AC(a6),a1          ; a1 -> TimeRequest für Threshold/Period
5FB6         move.l  a6,-(a7)             ; a6 retten
5FB8         movea.l 26(a6),a6            ; a6 := SysBase
5FBC         jsr     -1BC(a6)             ; ---> OpenDevice
5FCØ         movea.l (a7)+,a6             ; a6 wiederherstellen
5FC2         tst.l   dØ                   ; timer.device geöffnet?
5FC4         bne     FE6Ø5C               ; Nein: Fehler ->
```

```
5FC8          move.l  11C0(a6),11E8(a6)    ; io_Device und
5FCE          move.l  11C4(a6),11EC(a6)    ; io_Unit in GetSysTime-Request kopieren
5FD4          move.l  11C0(a6),1100C(a6)   ; io_Device und
5FDA          move.l  11C4(a6),1110(a6)    ; io_Unit in TimerEvent-Request kopieren
5FE0          lea     A5(pc),a0            ; a0 := FE6087 -> 'keyboard.device'
5FE4          moveq   #0,d0                ; d0 := 0 = UnitNumber
5FE6          moveq   #0,d1                ; d1 := 0 = Flags
5FE8          lea     117C(a6),a1          ; a1 -> Keyboard-ReadRequest
5FEC          move.l  a6,-(a7)             ; a6 retten
5FEE          movea.l 26(a6),a6            ; a6 := SysBase
5FF2          jsr     -1BC(a6)             ; ---> OpenDevice
5FF6          movea.l (a7)+,a6             ; a6 wiederherstellen
5FF8          tst.l   d0                   ; keyboard.device geöffnet?
5FFA          bne     FE6054               ; Nein: Fehler ->
5FFE          lea     10B2(a6),a0          ; a0 -> InputHandler-Liste
6002          move.l  a0,(a0)              ; Macro NewList
6004          addq.l  #4,(a0)              ; :
6006          clr.l   4(a0)                ; :
600A          move.l  a0,8(a0)             ; :
600E          lea     56(a6),a1            ; a1 -> Task
6012          lea     B2(a6),a0            ; a0 -> TaskStack-Untergrenze
6016          move.l  a0,3A(a1)            ; tc_SPLower := a0
601A          move.w  #7FF,d0              ; Zähler für 1 kWorte setzen
601E          move.w  #A6A6,d1             ; d1 := Füllkode
6022 FE6022   move.w  d1,(a0)+             ; TaskStack mit Kode $A6 füllen
6024          dbra    d0,FE6022            ; wiederholen bis Stack gefüllt ->
6028          move.l  a0,3E(a1)            ; tc_SPUpper := a0
602C          movea.l 26(a6),a2            ; a2 := SysBase
6030          move.l  a2,-(a0)             ; SysBase auf TaskStack
6032          move.l  a6,-(a0)             ; TaskBase auf TaskStack
6034          move.l  a0,36(a1)            ; tc_SPReg := a0
6038          lea     29A(pc),a2           ; a2 := FE62D4 -> InitialPC für Task
603C          moveq   #0,d0                ; d0 := 0
603E          movea.l d0,a3                ; a3 := 0 (FinalPC)
6040          move.l  a6,-(a7)             ; a6 retten
6042          movea.l 26(a6),a6            ; a6 := SysBase
6046          jsr     -11A(a6)             ; ---> AddTask
604A          movea.l (a7)+,a6             ; a6 wiederherstellen
604C          move.l  a6,d0                ; d0 := a6 -> Input.device
```

```
604E FE604E  movem.l (a7)+,a2-a3/a6      ; Register wiederherstellen
6052         rts
6054
6054 ;-------------------------------   Fehlerausgang
6054
6054 FE6054  lea     10F8(a6),a1         ; a1 -> IORequest
6058         jsr     -1C2(a6)            ; ---> CloseDevice (Panic!! a6 falsch!)
605C FE605C  movea.l a6,a1               ; a1 := a6 = DevBase
605E         move.w  10(a6),d0           ; d0 := NegSize
6062         suba.w  d0,a1               ; a1 -> Anfang der Sprungliste
6064         add.w   12(a6),d0           ; d0 := NegSize+PosSize
6068         ext.l   d0                  ; Auf Langwort erweitern
606A         move.l  a6,-(a7)            ; a6 retten
606C         movea.l 26(a6),a6           ; a6 := SysBase
6070         jsr     -D2(a6)             ; ---> FreeMem
6074         movea.l (a7)+,a6            ; a6 wiederherstellen
6076         moveq   #0,d0               ; d0 := 0
6078         bra.s   FE604E              ; ---> Ausgang
607A
607A ;-------------------------------   Device-Namen
607A
607A FE607A  DC.B    'timer.device',0
6087
6087 FE6087  DC.B    'keyboard.device',0,0
6098
6098 ;-------------------------------   Tabelle für InitStruct
6098
6098 FE6098  DC.W    E000                ; InitByte
609A         DC.W    8                   ; ln_Type
609C         DC.B    3                   ; = nt_Device
609D         DC.B    0
609E         DC.W    C000                ; InitLong
60A0         DC.W    A                   ; lib_Name
60A2         DC.L    FE50E0              ; -> 'input.device'
60A6         DC.W    E000                ; InitByte
60A8         DC.W    E                   ; lib_Flags
60AA         DC.B    6                   ; = Changed, Sumused
60AB         DC.B    0
60AC         DC.W    D000                ; InitWord
60AE         DC.W    14                  ; Version
```

```
60B0        DC.W    21              ; = 33
60B2        DC.W    D000            ; InitWord
60B4        DC.W    16              ; Revision
60B6        DC.W    67              ; = 103
60B8        DC.W    C000            ; InitLong
60BA        DC.W    2A
60BC        DC.L    FE6142          ; -> Kommando-Tabelle
60C0        DC.W    C000            ; InitLong
60C2        DC.W    2E
60C4        DC.L    FE6186          ; -> MsgPort-Nummern
60C8        DC.W    D000            ; InitWord
60CA        DC.W    32
60CC        DC.W    11              ; Letztes Kommando + 1
60CE        DC.W    E000            ; InitByte
60D0        DC.W    3C              ; ln_Type
60D2        DC.B    4               ; = nt_MsgPort
60D3        DC.B    0
60D4        DC.W    C000            ; InitLong
60D6        DC.W    3E              ; ln_Name
60D8        DC.L    FE50E0          ; -> 'input.device'
60DC        DC.W    E000            ; InitByte
60DE        DC.W    5E              ; ln_Type
60E0        DC.B    1               ; = nt_Task
60E1        DC.B    0
60E2        DC.W    E000            ; InitByte
60E4        DC.W    5F              ; ln_Pri
60E6        DC.B    14              ; = 20
60E7        DC.B    0
60E8        DC.W    C000            ; InitLong
60EA        DC.W    60              ; ln_Name
60EC        DC.L    FE50E0          ; -> 'input.device'
60F0        DC.W    C000            ; InitLong
60F2        DC.W    11FC            ; Threshold (secs)
60F4        DC.L    0               ; = 0
60F8        DC.W    C000            ; InitLong
60FA        DC.W    1200            ; Threshold (micro)
60FC        DC.L    0C3500          ; = 800000
6100        DC.W    C000            ; InitLong
6102        DC.W    1204            ; Period (secs)
6104        DC.L    0               ; = 0
```

```
6108          DC.W    C000                ; InitLong
610A          DC.W    1208                ; Period (micro)
610C          DC.L    0186A0              ; = 100000
6110          DC.W    D000                ; InitWord
6112          DC.W    1212                ; Repeat-Kode (Tastenwiederholung)
6114          DC.W    -1                  ; Wiederholung gesperrt
6116          DC.W    E000                ; InitByte
6118          DC.W    1229                ; MType
611A          DC.B    1                   ; = 1 = Maus
611B          DC.B    0
611C          DC.W    D000                ; InitWord
611E          DC.W    122A                ; gpt_Keys
6120          DC.W    3                   ; = 3 (DownKeys & UpKeys)
6122          DC.W    C000                ; InitLong
6124          DC.W    122E                ; gpt_XDelta, gpt_YDelta
6126          DC.L    010001              ; = 1 / 1
612A          DC.W    E000                ; InitByte
612C          DC.W    3D                  ; ln_Pri im MsgPort (Flags)
612E          DC.B    2                   ; = 2 = Gameport-Änderungsflag
612F          DC.B    0
6130          DC.W    0
6132
6132 ;--------------------------------   Vektoren für Sprungliste
6132
6132 FE6132   DC.W    -1
6134          DC.W    0088                ; FE61BA  Open
6136          DC.W    F1C4                ; FE52F6  Close
6138          DC.W    007C                ; FE61AE  Expunge
613A          DC.W    F1C2                ; FE52F4  Extfunct
613C          DC.W    F02A                ; FE515C  BeginIO
613E          DC.W    F0EE                ; FE5220  AbortIO
6140          DC.W    -1
6142
6142 ;--------------------------------   Adressentabelle für IO-Kommandos
6142
6142 FE6142   DC.L    FE5314              ; $00 ind_Invalid
6146          DC.L    FE6198              ; $01 ind_Reset
614A          DC.L    FE5388              ; $02 ind_Read
614E          DC.L    FE5388              ; $03 ind_Write
6152          DC.L    FE5388              ; $04 ind_Update
```

```
6156          DC.L    FE61DØ              ; $Ø5 ind_Clear
615A          DC.L    FE532A              ; $Ø6 ind_Stop
615E          DC.L    FE61D2              ; $Ø7 ind_Start
6162          DC.L    FE5366              ; $Ø8 ind_Flush
6166          DC.L    FE61A6              ; $Ø9 ind_AddHandler
616A          DC.L    FE61A6              ; $ØA ind_RemHandler
616E          DC.L    FE61A6              ; $ØB ind_WriteEvent
6172          DC.L    FE624A              ; $ØC ind_SetThresh
6176          DC.L    FE625C              ; $ØD ind_SetPeriod
617A          DC.L    FE626A              ; $ØE ind_SetMPort
617E          DC.L    FE6284              ; $ØF ind_SetMType
6182          DC.L    FE629E              ; $1Ø ind_SetMTrig
6186
6186 ;--------------------------------   Kommando-Flags
6186
6186 FE6186   DC.B    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,Ø,Ø,Ø,Ø,-1,-1,-1,-1,-1,-1
6197
6197          DC.B    Ø
6198
6198 ;--------------------------------   ind_Reset
6198
6198 FE6198   bsr     FE532A              ; ---> Stop
619C          bsr     FE5366              ; ---> Flush
61AØ          bsr     FE61D2              ; ---> Start
61A4          rts
61A6
61A6 ;--------------------------------   AddHandler, RemHandler, WriteEvent
61A6
61A6 FE61A6   bclr.b  #Ø,1E(a1)           ; QuickIO-Flag löschen
61AC          rts
61AE
61AE ;--------------------------------   Expunge
61AE
61AE FE61AE   tst.w   2Ø(a6)              ; Open-Zähler = Ø?
61B2          bne     FE52EC              ; Nein: ->
61B6          bra     FE52EC              ; ---> (Sehr merkwürdig!)
61BA
61BA ;--------------------------------   Open
61BA
61BA FE61BA   lea     34(a6),aØ           ; aØ -> MsgPort im input.device
```

```
61BE          move.l  a0,18(a1)           ; als io_Unit speichern
61C2          bclr.b  #3,E(a6)            ; DelExp-Flag löschen
61C8          addq.w  #1,20(a6)           ; Open-Zähler erhöhen
61CC          rts
61CE
61CE          DC.W    0
61D0
61D0 ;--------------------------------   ind_Clear
61D0
61D0 FE61D0   rts
61D2
61D2 ;--------------------------------   ind_Start
61D2
61D2 FE61D2   move.l  a1,-(a7)            ; a1 retten
61D4          movea.l 18(a1),a0           ; a0 -> io_Unit
61D8          bclr.b  #0,9(a0)            ; Stop-Flag löschen
61DE          beq.s   FE61F8              ; War schon gelöscht: ->
61E0          lea     56(a6),a1           ; a1 -> Task
61E4          move.b  F(a0),d1            ; d1 := mp_SigBit
61E8          moveq   #0,d0               ; d0 := 0
61EA          bset.l  d1,d0               ; Signalbit in d0 setzen
61EC          move.l  a6,-(a7)            ; a6 retten
61EE          movea.l 26(a6),a6           ; a6 := SysBase
61F2          jsr     -144(a6)            ; ---> Signal
61F6          movea.l (a7)+,a6            ; a6 wiederherstellen
61F8 FE61F8   movea.l (a7)+,a1            ; a1 wiederherstellen
61FA          bra.s   FE6256              ; ---> IORequest beantworten
61FC
61FC ;--------------------------------   ind_AddHandler
61FC
61FC FE61FC   move.l  a1,-(a7)            ; a1 retten
61FE          movea.l 28(a1),a1           ; a1 := io_Data -> InputHandler-Struktur
6202          lea     10B2(a5),a0         ; a0 -> HandlerList-Header
6206          jsr     -10E(a6)            ; ---> Enqueue
620A          movea.l (a7)+,a1            ; a1 wiederherstellen
620C          move.l  a6,-(a7)            ; a6 retten
620E          movea.l a5,a6               ; a6 := a5 -> InputDevice
6210          bsr.s   FE6256              ; ---> IORequest beantworten
```

```
6212          movea.l (a7)+,a6            ; a6 wiederherstellen
6214          rts
6216
6216 ;--------------------------------   ind_RemHandler
6216
6216 FE6216   move.l  a1,-(a7)            ; a1 retten
6218          movea.l 28(a1),a1           ; a1 := io_Data -> InputHandler-Struktur
621C          movea.l (a1),a0             ; Macro Remove
621E          movea.l 4(a1),a1            ; :
6222          move.l  a0,(a1)             ; :
6224          move.l  a1,4(a0)            ; :
6228          movea.l (a7)+,a1            ; a1 wiederherstellen
622A          move.l  a6,-(a7)            ; a6 retten
622C          movea.l a5,a6               ; a6 := a5 -> InputDevice
622E          bsr.s   FE6256              ; ---> IORequest beantworten
6230          movea.l (a7)+,a6            ; a6 wiederherstellen
6232          rts
6234
6234 ;--------------------------------   ind_WriteEvent
6234
6234 FE6234   move.l  a1,-(a7)            ; a1 retten
6236          movea.l 28(a1),a3           ; a3 := io_Data -> InputEvent-Kette
623A          bsr     FE66B2              ; ---> InputHandler rufen
623E          movea.l (a7)+,a1            ; a1 wiederherstellen
6240          move.l  a6,-(a7)            ; a6 retten
6242          movea.l a5,a6               ; a6 := a5 -> InputDevice
6244          bsr.s   FE6256              ; ---> IORequest beantworten
6246          movea.l (a7)+,a6            ; a6 wiederherstellen
6248          rts
624A
624A ;--------------------------------   ind_SetThresh
624A
624A FE624A   move.l  20(a1),11FC(a6)     ; Threshold Sekunden
6250          move.l  24(a1),1200(a6)     ; Threshold Mikrosekunden
6256 FE6256   moveq   #0,d0               ; d0 := ok-Kode
6258          bra     FE5226              ; ---> IORequest beantworten
625C
625C ;--------------------------------   ind_SetPeriod
625C
625C FE625C   move.l  20(a1),1204(a6)     ; Period Sekunden
```

```
6262          move.l  24(a1),1208(a6)     ; Period Mikrosekunden
6268          bra.s   FE6256              ; ---> IORequest beantworten
626A
626A ;--------------------------------   ind_SetMPort
626A
626A FE626A   move.l  24(a1),d0           ; d0 := io_Length
626E          subq.w  #1,d0               ; - 1
6270          bcs     FE62CC              ; < 0: Fehler ->
6274          movea.l 28(a1),a0           ; a0 -> MPort (0 oder 1)
6278          move.b  (a0),1228(a6)       ; MPort in InputDevice kopieren
627C          bset.b  #1,3D(a6)           ; Gameport-Änderungsflag setzen
6282          rts
6284
6284 ;--------------------------------   ind_SetMType
6284
6284 FE6284   move.l  24(a1),d0           ; d0 := io_Length
6288          subq.w  #1,d0               ; - 1
628A          bcs     FE62CC              ; < 0: Fehler ->
628E          movea.l 28(a1),a0           ; a0 -> MType (-1,0,1,2,3)
6292          move.b  (a0),1229(a6)       ; MType in InputDevice kopieren
6296          bset.b  #1,3D(a6)           ; Gameport-Änderungsflag setzen
629C          rts
629E
629E ;--------------------------------   ind_SetMTrig
629E
629E FE629E   move.l  24(a1),d0           ; d0 := io_Length
62A2          subq.w  #8,d0               ; - 8 (Länge von GamePortTrigger)
62A4          bcs.s   FE62CC              ; < 0: Fehler ->
62A6          move.l  a1,-(a7)            ; a1 retten
62A8          lea     122A(a6),a0         ; a0 -> MTrig
62AC          movea.l 28(a1),a1           ; a1 := io_Data -> GamePortTrigger
62B0          moveq   #3,d0               ; Zähler für 4 Worte setzen
62B2 FE62B2   move.w  (a1)+,(a0)+         ; Triggerwort in InputDevice kopieren
62B4          dbra    d0,FE62B2           ; 3 mal wiederholen ->
62B8          bset.b  #1,3D(a6)           ; Gameport-Änderungsflag setzen
62BE          movea.l (a7)+,a1            ; a1 wiederherstellen
62C0          moveq   #8,d0               ; d0 := 8
62C2          move.l  d0,20(a1)           ; io_Actual := d0
```

```
62C6          moveq   #0,d0                ; d0 := ok-Kode
62C8          bra     FE5226               ; ---> IORequest beantworten
62CC
62CC ;--------------------------------   Fehlerausgang
62CC
62CC FE62CC   moveq   #-3,d0               ; d0 := ioerr_NoCmd
62CE          bra     FE5226               ; ---> IORequest beantworten
62D2
62D2          DC.W    0
62D4
62D4 ;--------------------------------   Input-Task
62D4
62D4 FE62D4   movea.l 4(a7),a5             ; a5 -> InputDevice
62D8          movea.l 8(a7),a4             ; a4 := SysBase
62DC          movea.l a5,a6                ; a6 := a5 -> InputDevice
62DE          lea     34(a5),a0            ; a0 -> MsgPort im InputDevice
62E2          bsr     FE538C               ; ---> Signal zuweisen (CommandPort)
62E6          moveq   #0,d7                ; d7 := 0
62E8          bset.l  d0,d7                ; Signalbit in d7 setzen
62EA          move.l  d7,d6                ; d6 := d7
62EC          lea     10C0(a5),a0          ; a0 -> ReplyPort
62F0          move.l  a0,11BA(a5)          ; in Thresh/Period AddRequest,
62F4          move.l  a0,11E2(a5)          ; in GetSysTime Request,
62F8          move.l  a0,1106(a5)          ; in Timer AddRequest,
62FC          move.l  a0,1144(a5)          ; in Gameport Input Request und
6300          move.l  a0,118A(a5)          ; in Keyboard Input Request eintragen
6304          bsr     FE538C               ; ---> Signal zuweisen (ReplyPort)
6308          bset.l  d0,d7                ; Signalbit in d7 setzen
630A          movea.l a4,a6                ; a6 := a4 = SysBase
630C          move.l  d7,d0                ; d0 := d7 = Wecksignal
630E          jsr     -13E(a6)             ; ---> Wait
6312          move.l  d0,d2                ; d2 := d0 = empfangenes Signal
6314          bsr.s   FE6358               ; ---> Gameport-ReadRequest senden
6316          bsr     FE65D2               ; ---> TimerEvent-Request senden
631A          bsr     FE6612               ; ---> Keyboard-ReadRequest senden
631E          bra     FE658E               ; ---> Kommandoverteiler
6322
6322 FE6322   DC.B    'gameport.device',0
6332
```

```
6332 ;--------------------------------   Gameport zuordnen
6332
6332 FE6332  tst.l   114A(a5)           ; gameport.device aktiv?
6336         beq.s   FE6358             ; nein: ->
6338         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
633C         move.l  a6,-(a7)           ; a6 retten
633E         movea.l 14(a1),a6          ; a6 -> gameport.device
6342         jsr     -24(a6)            ; ---> AbortIO
6346         movea.l (a7)+,a6           ; a6 wiederherstellen
6348         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
634C         jsr     -1DA(a6)           ; ---> WaitIO
6350         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
6354         jsr     -1C2(a6)           ; ---> CloseDevice
6358 FE6358  bclr.b  #1,3D(a5)          ; Änderungsflag löschen
635E         lea     -3E(pc),a0         ; a0 := FE6322 -> 'gameport.device'
6362         moveq   #0,d0              ; d0 := 0
6364         move.b  1228(a5),d0        ; d0 := MPort (0 oder 1) = Unit
6368         moveq   #0,d1              ; d1 := 0 = Flags
636A         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
636E         jsr     -1BC(a6)           ; ---> OpenDevice
6372         tst.l   d0                 ; Device geöffnet?
6374         bne     FE63C4             ; nein: Device nicht vorhanden ->
6378         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
637C         move.w  #B,1C(a1)          ; io_Command := gpd_SetCType
6382         move.l  #1,24(a1)          ; io_Length := 1
638A         lea     1229(a5),a0        ; a0 -> MType
638E         bsr.s   FE63AC             ; ---> Kommando ausführen
6390         lea     1136(a5),a1        ; a1 -> Gameport-ReadRequest
6394         move.w  #D,1C(a1)          ; io_Command := gpd_SetTrigger
639A         move.l  #8,24(a1)          ; io_Length := 8
63A2         lea     122A(a5),a0        ; a0 -> MTrig
63A6         bsr.s   FE63AC             ; ---> Kommando ausführen
63A8         bra     FE65F2             ; ---> gpd_ReadEvent-Request senden
63AC
63AC FE63AC  move.l  a0,28(a1)          ; io_Data := a0
63B0         move.b  #1,1E(a1)          ; io_Flags := QuickIO
63B6         bsr     FE6630             ; ---> BeginIO rufen
63BA         lea     1136(a5),a1        ; a1 -> Gameport IORequest
63BE         jsr     -1DA(a6)           ; ---> WaitIO
63C2         rts
```

```
63C4
63C4 FE63C4  clr.l   114A(a5)          ; io_Device := Ø
63C8         rts
63CA
63CA ;------ Task-Hauptschleife
63CA
63CA FE63CA  btst.b  #Ø,3D(a5)         ; Stop-Flag gesetzt?
63DØ         beq.s   FE63DA            ; nein: ->
63D2         move.l  d6,dØ             ; dØ := d6 = CmdPort-Signal
63D4         jsr     -13E(a6)          ; ---> Wait
63D8         bra.s   FE63FC            ; --->
63DA
63DA FE63DA  move.l  d7,dØ             ; dØ := d7 = CmdPort- u.ReplyPort-Signal
63DC         jsr     -13E(a6)          ; ---> Wait
63EØ         btst.b  #Ø,3D(a5)         ; Stop-Flag gesetzt?
63E6         beq.s   FE63FC            ; nein: ->
63E8         move.l  dØ,d2             ; d2 := dØ = Wecksignal
63EA         and.l   d6,d2             ; CmdPort-Signal isolieren
63EC         move.l  d7,d1             ; d1 := CmdPort- & ReplyPort-Signal
63EE         eor.l   d6,d1             ; CmdPort-Signal entfernen
63FØ         move.l  d1,d5             ; d5 := ReplyPort-Signal
63F2         and.l   dØ,d5             ; Wecksignal = ReplyPort-Signal?
63F4         beq.s   FE63FE            ; nein: ->
63F6         jsr     -132(a6)          ; ---> SetSignal in tc_SigRecvd
63FA         bra.s   FE63FE            ; --->
63FC
63FC FE63FC  move.l  dØ,d2             ; d2 := dØ = Wecksignal
63FE FE63FE  move.b  1ØCF(a5),dØ       ; dØ := mp_SigBit im ReplyPort
6402         ext.w   dØ                ; auf Wort erweitern
6404         btst.l  dØ,d2             ; Ist es das Wecksignal?
6406         beq     FE6582            ; nein: ->
640A FE640A  lea     1ØC0(a5),aØ       ; aØ -> ReplyPort
640E         jsr     -174(a6)          ; ---> GetMsg
6412         tst.l   dØ                ; Message vorhanden?
6414         beq     FE6582            ; nein: Kommandoverteiler ->
6418         lea     1ØF8(a5),a1       ; a1 -> TimerEvent-Request
641C         cmp.l   a1,dØ             ; Message = TimerEvent-Request?
641E         beq.s   FE643C            ; ja: TimerEvent erzeugen ->
6420         lea     1136(a5),a1       ; a1 -> Gameport-ReadRequest
6424         cmp.l   a1,dØ             ; Message = Gameport-ReadRequest?
```

```
6426          beq.s   FE645E              ; ja: GameportEvent erzeugen ->
6428          lea     11AC(a5),a1         ; a1 -> Threshold/Period-TimeRequest
642C          cmp.l   a1,dØ               ; Message = Thresh/Period-TimeRequest?
642E          beq     FE6546              ; ja: Tastenwiederholung ->
6432          lea     117C(a5),a1         ; a1 -> Keyboard-ReadRequest
6436          cmp.l   a1,dØ               ; Message = Keyboard-ReadRequest?
6438          beq.s   FE649A              ; ja: KeyboardEvent erzeugen
643A          bra.s   FE640A              ; ---> Loop
643C
643C ;------ TimerEvent erzeugen
643C
643C FE643C   lea     1ØE2(a5),a3         ; a3 -> TimerEvent
6440          move.l  18C(pc),4(a3)       ; Event-Parameter von FE65CE kopieren
6446          clr.l   A(a3)               ; ie_EventAddress := Ø
644A          bsr     FE663E              ; ---> GetSysTime-Request ausführen
644E          movem.l d3-d4,E(a3)         ; TimeStamp in Event eintragen
6454          bsr     FE66B2              ; ---> InputHandler rufen
6458          bsr     FE65D2              ; ---> TimerEvent-Request senden
645C          bra.s   FE640A              ; ---> Loop
645E
645E ;------ GameportEvent erzeugen
645E
645E FE645E   btst.b  #1,3D(a5)           ; Gameport-Änderungsflag gesetzt?
6464          bne.s   FE6496              ; ja: Fertig ->
6466          lea     112Ø(a5),a3         ; a3 -> GameportEvent
646A          bsr     FE663E              ; ---> GetSysTime-Request ausführen
646E          movem.l d3-d4,E(a3)         ; TimeStamp in Event eintragen
6474          move.w  8(a3),dØ            ; dØ := ie_Qualifier
6478          move.w  1222(a5),d1         ; d1 := letzter Qualifier-Status
647C          andi.w  #FØØØ,dØ            ; dØ: Maus-Qualifier-Bits isolieren
6480          andi.w  #FFF,d1             ; d1: Übrige Qualifier-Bits isolieren
6484          or.w    dØ,d1               ; Maus-Qualifier-Bits einodern
6486          move.w  d1,8(a3)            ; Neuen Qualifier-Status im Event und
648A          move.w  d1,1222(a5)         ; im input.device speichern
648E          bsr     FE66B2              ; ---> InputHandler rufen
6492          bsr     FE65F2              ; ---> gpd_ReadEvent-Request senden
6496 FE6496   bra     FE640A              ; ---> Loop
649A
```

```
649A ;------ KeyboardEvent erzeugen
649A
649A FE649A  lea     1166(a5),a3          ; a3 -> KeyboardEvent
649E         bsr     FE663E               ; ---> GetSysTime-Request ausführen
64A2         movem.l d3-d4,E(a3)          ; TimeStamp in Event eintragen
64A8         move.w  8(a3),dØ             ; dØ := ie_Qualifier
64AC         move.w  1222(a5),d1          ; d1 := letzter Qualifier-Status
64BØ         andi.w  #7FF,dØ              ; dØ: Keyboard-Qualifier-Bits isolieren
64B4         andi.w  #F8ØØ,d1             ; d1: Übrige Qualifier-Bits isolieren
64B8         or.w    dØ,d1                ; Keyboard-Qualifier-Bits einodern
64BA         move.w  d1,8(a3)             ; Neuen Qualifier-Status im Event und
64BE         move.w  d1,1222(a5)          ; im input.device speichern
64C2         move.w  6(a3),dØ             ; dØ := ie_Code
64C6         btst.l  #7,dØ                ; Bit 7 (Upkey-Flag) gesetzt?
64CA         bne.s   FE65ØC               ; ja: Taste losgelassen ->
64CC         cmpi.w  #6Ø,dØ               ; Shift, Ctrl, Alt, Amiga?
64DØ         bge.s   FE652A               ; ja: ->
64D2         move.l  1224(a5),A(a3)       ; Letzten Kode u. Qualifier in Event
64D8         move.w  1224(a5),1226(a5)    ; und Kodestack speichern
64DE         move.b  dØ,1224(a5)          ; Neuen Kode und
64E2         move.b  d1,1225(a5)          ; Qualifier (Ø,...,7) in Kodestack
64E6         tst.w   1212(a5)             ; Tastenwiederholung aktiv?
64EA         bmi.s   FE64FØ               ; nein: ->
64EC         bsr     FE6698               ; ---> AbortIO ausführen
64FØ FE64FØ  moveq   #8,d1                ; d1 := 8 = Zähler für Event-Kopie
64F2         lea     121Ø(a5),a1          ; a1 -> RepeatEvent
64F6         lea     4(a3),aØ             ; aØ -> Keyboard-ReadEvent
64FA FE64FA  move.w  (aØ)+,(a1)+          ; Wort in RepeatEvent kopieren
64FC         dbra    d1,FE64FA            ; 8 mal wiederholen ->
65ØØ         ori.w   #2ØØ,1214(a5)        ; Repeat-Qualifier setzen
6506         bsr     FE666Ø               ; ---> Threshold-TimeRequest senden
65ØA         bra.s   FE653A               ; ---> Abschlup
65ØC
65ØC FE65ØC  bclr.l  #7,dØ                ; Upkey-Bit löschen
651Ø         cmp.w   1212(a5),dØ          ; Tastenwiederholung aktiv?
6514         beq.s   FE652A               ; ja: Wiederholung abbrechen ->
6516         move.w  1214(a5),dØ          ; dØ := Qualifier im RepeatEvent
651A         andi.w  #5FF,dØ              ; Repeat-Flag löschen
651E         move.w  1222(a5),d1          ; d1 := letzter Qualifier
6522         andi.w  #7FF,d1              ; Keyboard-Bits isolieren
```

```
6526          cmp.w   dØ,d1               ; Qualifier unverändert?
6528          beq.s   FE653A              ; ja: ->
652A FE652A   tst.w   1212(a5)            ; Tastenwiederholung aktiv?
652E          bmi.s   FE653A              ; nein: ->
653Ø          move.w  #-1,1212(a5)        ; Wiederholungskode löschen
6536          bsr     FE6698              ; ---> AbortIO ausführen
653A FE653A   bsr     FE66B2              ; ---> InputHandler rufen
653E          bsr     FE6612              ; ---> kbd_ReadEvent-Request senden
6542          bra     FE640A              ; ---> Loop
6546
6546 ;------ Tastenwiederholung
6546
6546 FE6546   lea     120C(a5),a3         ; a3 -> RepeatEvent
654A          tst.w   6(a3)               ; Tastenwiederholung aktiv?
654E          bmi     FE640A              ; nein: Loop ->
6552          bsr     FE663E              ; ---> GetSysTime ausführen
6556          movem.l d3-d4,E(a3)         ; TimeStamp in Event eintragen
655C          move.w  8(a3),dØ            ; dØ := ie_Qualifier
6560          move.w  1222(a5),d1         ; d1 := letzter Qualifier
6564          andi.w  #7FF,dØ             ; dØ: Keyboard-Bits isolieren
6568          andi.w  #F8ØØ,d1            ; d1: Übrige Bits isolieren
656C          or.w    dØ,d1               ; d1 := neuer Qualifier
656E          move.w  d1,8(a3)            ; in Event eintragen
6572          move.w  d1,1222(a5)         ; und im input.device speichern
6576          bsr     FE66B2              ; ---> InputHandler rufen
657A          bsr     FE667C              ; ---> Period-TimeRequest senden
657E          bra     FE640A              ; ---> Loop
6582
6582 ;------ Kommando-Verteiler
6582
6582 FE6582   btst.b  #1,3D(a5)           ; Gameport-Änderungsflag gesetzt?
6588          beq.s   FE658E              ; nein: ->
658A          bsr     FE6332              ; ---> Gameport zuordnen
658E FE658E   move.b  43(a5),d1           ; d1 := mp_SigBit im CommandPort
6592          ext.w   d1                  ; auf Wort erweitern
6594          btst.l  d1,d2               ; Message am CommandPort?
6596          beq     FE63CA              ; nein: Loop ->
659A          movea.l 48(a5),a1           ; a1 := MsgList.lh_Head am CmdPort
659E          tst.l   (a1)                ; Ende der Liste?
65AØ          beq     FE63CA              ; ja: Loop ->
```

```
65A4          move.w  1C(a1),d0           ; d0 := io_Command
65A8          bclr.b  #0,1E(a1)           ; io_Flags.QuickIO löschen
65AE          subi.w  #9,d0               ; d0 := io_Command - 9
65B2          lsl.w   #2,d0               ; mal 4 ergibt Tabellenoffset
65B4          lea     C(pc),a0            ; a0 := FE65C2
65B8          movea.l 0(a0,d0.w),a0       ; a0 := Adresse aus Tabelle
65BC          jsr     (a0)                ; ---> Routine rufen
65BE          bra     FE63CA              ; ---> Loop
65C2
65C2 FE65C2   DC.L    FE61FC              ; ind_AddHandler
65C6          DC.L    FE6216              ; ind_RemHandler
65CA          DC.L    FE6234              ; ind_WriteEvent
65CE
65CE ;------ Parameter für TimerEvent
65CE
65CE FE65CE   DC.B    6,0,0,0             ; ie_Class, ie_Subclass, ie_Code
65D2
65D2 ;------ TimerEvent-Request senden
65D2
65D2 FE65D2   lea     10F8(a5),a1         ; a1 -> TimerEvent Request
65D6          move.w  #9,1C(a1)           ; io_Command := tr_AddRequest
65DC          clr.b   1E(a1)              ; io_Flags := 0
65E0          move.l  #0,20(a1)           ; tr_Secs := 0
65E8          move.l  #0186A0,24(a1)      ; tr_Micro := 100000
65F0          bra.s   FE6630              ; ---> BeginIO rufen
65F2
65F2 ;------ gpd_ReadEvent-Request senden
65F2
65F2 FE65F2   lea     1136(a5),a1 ; a1 -> Gameport Request
65F6          move.w  #9,1C(a1)           ; io_Command := gpd_ReadEvent
65FC          clr.b   1E(a1)              ; io_Flags := 0
6600          move.l  #16,24(a1)          ; io_Length := 22 (Länge InputEvent)
6608          lea     1120(a5),a0 ; a0 -> GameportEvent
660C          move.l  a0,28(a1)           ; io_Data := a0
6610          bra.s   FE6630              ; ---> BeginIO rufen
6612
6612 ;------ kbd_ReadEvent-Request senden
6612
6612 FE6612   lea     117C(a5),a1         ; a1 -> Keyboard Request
6616          move.w  #9,1C(a1)           ; io_Command := kbd_ReadEvent
```

```
661C          clr.b   1E(a1)              ; io_Flags := 0
662Ø          move.l  #16,24(a1)          ; io_Length := 22 (Länge InputEvent)
6628          lea     1166(a5),aØ         ; aØ -> KeyboardEvent
662C          move.l  aØ,28(a1)           ; io_Data := aØ
663Ø FE6633   move.l  a6,-(a7)            ; a6 retten
6632          movea.l 14(a1),a6           ; a6 -> io_Device
6636          jsr     -1E(a6)             ; ---> BeginIO
663A          movea.l (a7)+,a6            ; a6 wiederherstellen
663C          rts
663E
663E ;------ Systemzeit holen
663E
663E FE663E   lea     11D4(a5),a1         ; a1 -> GetSysTime-Request
6642          move.w  #A,1C(a1)           ; io_Command := tr_GetSysTime
6648          move.b  #1,1E(a1)           ; io_Flags := QuickIO
664E          bsr.s   FE6633              ; ---> BeginIO rufen
6651          lea     11D4(a5),a1         ; a1 -> GetSysTime Request
6654          jsr     -1DA(a6)            ; ---> WaitIO
6658          movem.l 11F4(a5),d3-d4      ; d3 := tv_Secs, d4 := tv_Micro
665E          rts
6661
6661 ;------ Threshold-TimeRequest senden
6661
6661 FE6661   lea     11AC(a5),a1         ; a1 -> Threshold/Period Request
6664          move.w  #9,1C(a1)           ; io_Command := tr_AddRequest
666A          clr.b   1E(a1)              ; io_Flags := Ø
666E          move.l  11FC(a5),2Ø(a1)     ; tr_Secs := Threshold.Secs
6674          move.l  12ØØ(a5),24(a1)     ; tr_Micro := Threshold.Micro
667A          bra.s   FE6633              ; ---> BeginIO rufen
667C
667C ;------ Period-TimeRequest senden
667C
667C FE667C   lea     11AC(a5),a1         ; a1 -> Threshold/Period Request
6681          move.w  #9,1C(a1)           ; io_Command := tr_AddRequest
6686          clr.b   1E(a1)              ; io_Flags := Ø
668A          move.l  12Ø4(a5),2Ø(a1)     ; tr_Secs := Period.Secs
6691          move.l  12Ø8(a5),24(a1)     ; tr_Micro := Period.Micro
6696          bra.s   FE6633              ; ---> BeginIO rufen
6698
```

```
6698 ;------ TimeRequest für Threshold/Period abbrechen
6698
6698 FE6698  lea     11AC(a5),a1          ; a1 -> Threshold/Period Request
669C         move.l  a6,-(a7)             ; a6 retten
669E         movea.l 14(a1),a6            ; a6 -> io_Device
66A2         jsr     -24(a6)              ; ---> AbortIO
66A6         movea.l (a7)+,a6             ; a6 wiederherstellen
66A8         lea     11AC(a5),a1          ; a1 -> Threshold/Period Request
66AC         jsr     -1DA(a6)             ; ---> WaitIO
66BØ         rts
66B2
66B2 ;--------------------------------   InputHandler rufen
66B2
66B2 FE66B2  movea.l a3,aØ                ; aØ := a3 -> InputEvent
66B4         clr.l   Ø(a3)                ; ie_NextEvent := Ø
66B8         movea.l 1ØB2(a5),a4          ; a4 := lh_Head der InputHandler-Liste
66BC FE66BC  tst.l   (a4)                 ; Liste zu Ende?
66BE         beq.s   FE66D4               ; ja: Fertig ->
66CØ         movea.l 12(a4),a2            ; a2 := is_Code -> InputHandler-Routine
66C4         movea.l E(a4),a1             ; a1 := is_Data
66C8         movea.l (a4),a4              ; a4 -> nächster InputHandler
66CA         jsr     (a2)                 ; ---> InputHandler-Routine rufen
66CC         tst.l   dØ                   ; Event für nächsten Handler vorhanden?
66CE         beq.s   FE66D4               ; nein: Fertig ->
66DØ         movea.l dØ,aØ                ; aØ := dØ -> InputEvent
66D2         bra.s   FE66BC               ; ---> Loop
66D4
66D4 FE66D4  rts
66D6
66D6         DC.W    Ø
66D8
66D8 ************************************************************************
```

Console.device Datenbereich

```
-030 ************************** Console.device ******************************
-030
-030          jmp    FE71BC       ; ---> RawKeyConvert
-02A          jmp    FE6DEC       ; ---> CDInputHandler
-024          jmp    FE5220       ; ---> AbortIO
-01E          jmp    FE515C       ; ---> BeginIO
-018          jmp    FE52F4       ; ---> Extfunct
-012          jmp    FE6D62       ; ---> Expunge
-00C          jmp    FE716E       ; ---> Close
-006          jmp    FE7044       ; ---> Open
0000
0000 ;-------------------------   Library-Block
0000
0000 cdBase   DC.L   4694         ; ln_Succ
0004          DC.L   327C         ; ln_Pred
0008          DC.B   3            ; ln_Type (3 = Device)
0009          DC.B   0            ; ln_Pri
000A          DC.L   FE5128       ; ln_Name -> 'console.device'
000E
000E          DC.B   6            ; lib_Flags (Changed, Sumused)
000F          DC.B   0            ; lib_Pad
0010          DC.W   30           ; lib_NegSize
0012          DC.W   B8           ; lib_PosSize
0014          DC.W   21           ; lib_Version
0016          DC.W   67           ; lib_Revision
0018          DC.L   0            ; lib_IdString
001C          DC.L   A4E40000     ; lib_Sum
0020          DC.W   4            ; lib_OpenCnt
0022
0022 ;-------------------------   Verschiedene Parameter
0022
0022          DC.L   0            ; SegList
```

```
0026          DC.L   676          ; SysBase
002A          DC.L   FE6CEE       ; cd_CmdArray
002E          DC.L   FE6D22       ; cd_CmdFlags
0032          DC.W   D            ; cd_LastComm
0034
0034          DC.L   21FE         ; GfxBase
0038          DC.L   4914         ; IntuiBase
003C
003C ;-------------------------   ConsoleUnit-Liste
003C
003C    00    DC.L   30FB8        ; lh_Head
0040    04    DC.L   0            ; lh_Tail
0044    08    DC.L   1C7C0        ; lh_TailPred
0048    0C    DC.B   0            ; lh_Type
0049    0D    DC.B   0            ; lh_Pad
004A
004A ;-------------------------   Aktiv-Zeiger
004A
004A          DC.L   0            ; Aktive ConsoleUnit
004E          DC.L   37C28        ; Aktives Window
0052
0052 ;-------------------------   IORequest
0052
0052    00    DC.L   1912         ; ln_Succ
0056    04    DC.L   190E         ; ln_Pred
005A    08    DC.B   7            ; ln_Type
005B    09    DC.B   0            ; ln_Pri
005C    0A    DC.L   0            ; ln_Name
0060    0E    DC.L   18FA         ; mn_ReplyPort
0064    12    DC.W   0            ; mn_Length
0066    14    DC.L   327C         ; io_Device
006A    18    DC.L   32B0         ; io_Unit
006E    1C    DC.W   9            ; io_Command
0070    1E    DC.B   A            ; io_Flags
0071    1F    DC.B   0            ; io_Error
0072    20    DC.L   0            ; io_Actual
0076    24    DC.L   0            ; io_Length
007A    28    DC.L   463A         ; io_Data
007E    2C    DC.L   0            ; io_Offset
0082
```

```
0082 ;-------------------------   InputHandler-Struktur
0082
0082    00   DC.L   4332         ; ln_Succ
0086    04   DC.L   4AC6         ; ln_Pred
008A    08   DC.B   2            ; ln_Type = Interrupt
008B    09   DC.B   0            ; ln_Pri
008C    0A   DC.L   FE5128       ; ln_Name -> 'console.device'
0090    0E   DC.L   *0000        ; is_Data -> console.device
0094    12   DC.L   *-02A        ; is_Code -> jmp CDInputHandler
0098
0098 ;-------------------------   DefaultKeyMap-Struktur
0098
0098    00   DC.L   FE846C       ; km_LowKeyMapTypes
009C    04   DC.L   FE84E4       ; km_LowKeyMap
00A0    08   DC.L   FE844E       ; km_LowCapsable
00A4    0C   DC.L   FE845D       ; km_LoRepeatable
00A8    10   DC.L   FE84AC       ; km_HiKeyMapTypes
00AC    14   DC.L   FE85E4       ; km_HiKeyMap
00B0    18   DC.L   FE8456       ; km_HiCapsable
00B4    1C   DC.L   FE8465       ; km_HiRepeatable
00B8
00B8 ************************************************************************
```

Console.unit Datenbereich

```
0000 ************************* Console.Unit ******************************
0000
0000 ;-------------------------   Message Port
0000
0000          DC.L    1C908         ; ln_Succ
0004          DC.L    45F4          ; ln_Pred
0008          DC.B    4             ; ln_Type (4 = MsgPort)
0009          DC.B    0             ; ln_Pri
000A          DC.L    FE5128        ; ln_Name -> 'console.device'
000E
000E          DC.B    0             ; mp_Flags
000F          DC.B    0             ; mp_SigBit
0010          DC.L    0             ; mp_SigTask
0014          DC.L    26A5A         ; lh_Head (mp_MsgList)
0018          DC.L    0             ; lh_Tail
001C          DC.L    26A5A         ; lh_TailPred
0020          DC.B    5             ; lh_Type (5 = Message)
0021          DC.B    0             ; lh_Pad
0022
0022 ;-------------------------   Terminal-Parameter
0022
0022          DC.L    2FE48         ; cu_Window      Fenster-Zeiger
0026          DC.W    19            ; cu_XCP Zeichenposition
0028          DC.W    0             ; cu_YCP
002A          DC.W    4F            ; cu_XMax        Maximale Zeichenposition
002C          DC.W    16            ; cu_YMax
002E          DC.W    8             ; cu_XRSize      Zeichenraster-Größe
0030          DC.W    8             ; cu_YRSize
0032          DC.W    0             ; cu_XROrigin    Raster-Anfangspunkt
0034          DC.W    0             ; cu_YROrigin
0036          DC.W    27F           ; cu_XRExtant    Raster-Ausdehnung
0038          DC.W    B7            ; cu_YRExtant
003A          DC.W    270F          ; cu_XMinShrink Minimale Fenstergröße
```

```
003C        DC.W   270F          ; cu_YMinShrink
003E        DC.W   18            ; cu_XCCP        Kursorposition
0040        DC.W   1             ; cu_YCCP
0042
0042 ;-------------------------  Bereich für AskKeyMap und SetKeyMap
0042
0042        DC.L   28272         ; cu_KeyMapStruct
0046        DC.L   27A72
004A        DC.L   1F0E6
004E        DC.L   1F0F5
0052        DC.L   282B2
0056        DC.L   27E72
005A        DC.L   1F0EE
005E        DC.L   1F0FD
0062
0062 ;-------------------------  Tabulator-Marken
0062
0062        DS.W   50            ; cu_TabStops
0102
0102 ;-------------------------  Console Rastport Attribute
0102
0102        DC.B   FF            ; cu_Mask
0103        DC.B   1             ; cu_FgPen
0104        DC.B   0             ; cu_BgPen
0105        DC.B   0             ; cu_AOLPen
0106        DC.B   1             ; cu_DrawMode
0107        DC.B   0             ; cu_AreaPtSz
0108        DC.L   0             ; cu_AreaPtrn
010C        DC.B   CA,A,A,A,A,A,A,A      ; cu_Minterms
0114        DC.L   2390          ; cu_Font
0118        DC.B   0             ; cu_AlgoStyle
0119        DC.B   0             ; cu_TxFlags
011A        DC.W   8             ; cu_TxHeight
011C        DC.W   8             ; cu_TxWidth
011E        DC.W   6             ; cu_TxBaseLine
0120        DC.W   0             ; cu_TxSpacing
0122
0122 ;-------------------------  Console Modes und RawEvents-Schalter
0122
0122        DC.B   0,0,30        ; cu_Modes (PMB_AWM Bits)
```

```
0125          DC.B   0,0,0          ; cu_RawEvents (ieClass_Max Bits)
0128
0128 ;-------------------------   ConUnit Input-Puffer
0128
0128          DC.W   4B             ; BufferHead
012A          DC.W   4B             ; BufferTail
012C          DC.W   100            ; BufferLength
012E
012E          DS.B   100            ; Puffer (256 Bytes)
022E
022E ;-------------------------   Console-Rastport Zwischenspeicher
022E
022E          DC.L   2FF60          ; wd_RPort
0232          DC.W   0              ; rp_CPX
0234          DC.W   0              ; rp_CPY
0236          DC.B   FF             ; rp_Mask
0237          DC.B   0              ; rp_FgPen
0238          DC.B   0              ; rp_BgPen
0239          DC.B   FF             ; rp_AOLPen
023A          DC.B   1              ; rp_DrawMode
023B          DC.B   0              ; rp_AreaPtSz
023C          DC.L   0              ; rp_AreaPtrn
0240          DC.B   A,A,A,A,A,A,A,A      ; rp_Minterms
0248          DC.L   2390           ; rp_Font
024C          DC.B   0              ; rp_AlgoStyle
024D          DC.B   0              ; rp_TxFlags
024E          DC.W   8              ; rp_TxHeight
0250          DC.W   8              ; rp_TxWidth
0252          DC.W   6              ; rp_TxBaseline
0254          DC.W   0              ; rp_TxSpacing
0256
0256 ;-------------------------   Arbeitsbereich für cd_Write
0256
0256          DC.L   FE7A6C         ; Vector für Ausgabe
025A
025A ;------ Puffer für ESC-Sequenzen
025A
025A          DC.W   0              ; BufferHead
025C          DC.W   0              ; BufferTail
025E          DC.W   8              ; BufferLength
```

```
0260
0260          DS.B   8            ; Puffer (8 Bytes)
0268
0268 ;------ Puffer für CSI-Sequenzen
0268
0268          DC.W   1E           ; BufferHead
026A          DC.W   1E           ; BufferTail
026C          DC.W   32           ; BufferLength
026E
026E          DS.B   32           ; Puffer (50 Bytes)
02A0
02A0 ;------ Diverse Variablen
02A0
02A0          DC.W   0            ; Branch-Parameter
02A2          DC.B   -1           ; Kursor-Abschaltungs-Zähler
02A3          DC.B   -1           ; DisplayLock-Zähler
02A4          DC.B   2            ; Flags     Bit 0: Zeilenverkettung
02A5                              ;               1: Kursor eingeschaltet
02A5                              ;               2: Shrink-Flag
02A5                              ;               3: nicht benützt
02A5                              ;               4: Zeilenlänge gesetzt
02A5                              ;               5: Seitenlänge gesetzt
02A5                              ;               6: Links-Offset gesetzt
02A5                              ;               7: Top-Offset gesetzt
02A5
02A5 ************************************************************************
```

Console.device Routinen

```
66D8 ********************** Console.device ********************************
66D8
66D8 ;--------------------------------   Lock Display (Ausgabe sperren)
66D8
66D8 FE66D8  move.l  a5,-(a7)            ; a5 retten
66DA         addq.b  #1,2A3(a2)          ; Lock-Zähler erhöhen
66DE         bgt     FE677Ø              ; Display bereits gesperrt: ->
66E2         movea.l 22(a2),a1           ; a1 := cu_Window
66E6         movea.l 32(a1),a1           ; a1 := wd_RPort
66EA                                     ; Aktuelle Rastport-Parameter retten:
66EA         move.l  24(a1),232(a2)      ; rp_cp_x, rp_cp_y
66FØ         move.l  18(a1),236(a2)      ; rp_Mask, rp_FgPen, rp_BgPen, rp_AOLPen
66F6         move.w  1C(a1),23A(a2)      ; rp_DrawMode, rp_AreaPtSz
66FC         move.l  8(a1),23C(a2)       ; rp_AreaPtrn
67Ø2         move.l  28(a1),24Ø(a2)      ; rp_Minterms
67Ø8         move.l  2C(a1),244(a2)      ; rp_Minterms
67ØE         move.l  34(a1),248(a2)      ; rp_Font
6714         move.w  38(a1),24C(a2)      ; rp_AlgoStyle, rp_TxFlags
671A         move.l  3A(a1),24E(a2)      ; rp_TxHeight, rp_TxWidth
672Ø         move.l  3E(a1),252(a2)      ; rp_TxBaseLine, rp_TxSpacing
6726                                     ; ConUnit-Rastportattribute einsetzen:
6726         move.l  1Ø2(a2),18(a1)      ; rp_Mask, rp_FgPen, rp_BgPen, rp_AOLPen
672C         move.w  1Ø6(a2),1C(a1)      ; rp_DrawMode, rp_AreaPtSz
6732         move.l  1Ø8(a2),8(a1)       ; rp_AreaPtrn
6738         move.l  1ØC(a2),28(a1)      ; rp_Minterms
673E         move.l  11Ø(a2),2C(a1)      ; rp_Minterms
6744         move.l  114(a2),34(a1)      ; rp_Font
674A         move.w  118(a2),38(a1)      ; rp_AlgoStyle, rp_TxFlags
675Ø         move.l  11A(a2),3A(a1)      ; rp_TxHeight, rp_TxWidth
6756         move.l  11E(a2),3E(a1)      ; rp_TxBaseLine, rp_TxSpacing
675C         move.l  a1,22E(a2)          ; RastPort-Adresse speichern
```

```
676Ø          movea.l Ø(a1),a5            ; a5 := rp_Layer
6764          move.l  a6,-(a7)            ; a6 retten
6766          movea.l 34(a6),a6           ; a6 := GfxBase
676A          jsr     -1BØ(a6)            ; ---> LockLayerRom
676E          movea.l (a7)+,a6            ; a6 wiederherstellen
677Ø FE6778   bclr.b  #2,2A4(a2)          ; ShrinkFlag löschen
6776          beq.s   FE677C              ; war schon gelöscht: ->
6778          bsr     FE76BØ              ; ---> Ausgabebereich initialisieren
677C FE677C   movea.l (a7)+,a5            ; a5 wiederherstellen
677E          rts
6780
6780 ;--------------------------------   Unlock Display (Ausgabe freigeben)
6780
6780 FE6780   move.l  a5,-(a7)            ; a5 retten
6782          subq.b  #1,2A3(a2)          ; Lock-Zähler erniedrigen
6786          bpl     FE6810              ; Noch nicht -1: ->
678A          movea.l 22E(a2),a1          ; a1 := wd_RPort
678E                                      ; Aktuelle RP-Parameter übernehmen:
678E          move.l  18(a1),102(a2)      ; rp_Mask, rp_FgPen, rp_BgPen, rp_AOLPen
6794          move.w  1C(a1),106(a2)      ; rp_DrawMode, rp_AreaPtSz
679A          move.l  8(a1),108(a2)       ; rp_AreaPtrn
67AØ          move.l  28(a1),10C(a2)      ; rp_Minterms
67A6          move.l  2C(a1),110(a2)      ; rp_Minterms
67AC          move.l  34(a1),114(a2)      ; rp_Font
67B2          move.w  38(a1),118(a2)      ; rp_AlgoStyle, rp_TxFlags
67B8          move.l  3A(a1),11A(a2)      ; rp_TxHeight, rp_TxWidth
67BE          move.l  3E(a1),11E(a2)      ; rp_TxBaseLine, rp_TxSpacing
67C4                                      ; Rastport-Parameter wiederherstellen:
67C4          move.l  232(a2),24(a1)      ; rp_cp_x, rp_cp_y
67CA          move.l  236(a2),18(a1)      ; rp_Mask, rp_FgPen, rp_BgPen, rp_AOLPen
67DØ          move.w  23A(a2),1C(a1)      ; rp_DrawMode, rp_AreaPtSz
67D6          move.l  23C(a2),8(a1)       ; rp_AreaPtrn
67DC          move.l  240(a2),28(a1)      ; rp_Minterms
67E2          move.l  244(a2),2C(a1)      ; rp_Minterms
67E8          move.l  248(a2),34(a1)      ; rp_Font
67EE          move.w  24C(a2),38(a1)      ; rp_AlgoStyle, rp_TxFlags
67F4          move.l  24E(a2),3A(a1)      ; rp_TxHeight, rp_TxWidth
67FA          move.l  252(a2),3E(a1)      ; rp_TxBaseLine, rp_TxSpacing
68ØØ          movea.l Ø(a1),a5            ; a5 := rp_Layer
68Ø4          move.l  a6,-(a7)            ; a6 retten
```

```
6806          movea.l 34(a6),a6           ; a6 := GfxBase
680A          jsr     -1B6(a6)            ; ---> UnlockLayerRom
680E          movea.l (a7)+,a6            ; a6 wiederherstellen
6810 FE6810   movea.l (a7)+,a5            ; a5 wiederherstellen
6812          rts
6814
6814 ;--------------------------------   Byte in Puffer schreiben
6814
6814 FE6814   move.w  (a0),d1             ; d1 := BufferHead
6816          addq.w  #1,d1               ; + 1
6818          cmp.w   4(a0),d1            ; < Pufferlänge?
681C          blt.s   FE6820              ; ja: ->
681E          clr.w   d1                  ; d1 := 0
6820 FE6820   cmp.w   2(a0),d1            ; d1 = BufferTail?
6824          beq.s   FE6830              ; ja: Puffer voll ->
6826          move.b  d0,6(a0,d1.w)       ; Byte d0 in Puffer schreiben
682A          move.w  d1,(a0)             ; BufferHead setzen
682C          moveq   #0,d0               ; ok-Flag
682E          rts
6830
6830 FE6830   moveq   #1,d0               ; Flag 'Puffer voll'
6832          rts
6834
6834;--------------------------------   Wort in Puffer schreiben
6834
6834 FE6834   move.w  (a0),d1             ; d1 := BufferHead
6836          addq.w  #2,d1               ; + 2
6838          cmp.w   4(a0),d1            ; < Pufferlänge?
683C          blt.s   FE6840              ; ja: ->
683E          clr.w   d1                  ; d1 := 0
6840 FE6840   cmp.w   2(a0),d1            ; d1 = BufferTail?
6844          beq.s   FE6850              ; ja: Puffer voll ->
6846          move.w  d0,6(a0,d1.w)       ; Wort d0 in Puffer schreiben
684A          move.w  d1,(a0)             ; BufferHead setzen
684C          moveq   #0,d0               ; ok-Flag
684E          rts
6850
6850 FE6850   moveq   #2,d0               ; Flag 'Puffer voll'
6852          rts
6854
```

```
6854 ;--------------------------------   Byte aus Puffer lesen
6854
6854 FE6854  move.w  2(aØ),d1           ; d1 := BufferTail
6858         cmp.w   (aØ),d1            ; = BufferHead?
685A         beq.s   FE6872             ; ja: Puffer leer ->
685C         addq.w  #1,d1              ; + 1
685E         cmp.w   4(aØ),d1           ; < Pufferlänge?
6862         blt.s   FE6866             ; ja: ->
6864         clr.w   d1                 ; d1 := Ø
6866 FE6866  moveq   #Ø,dØ              ; dØ := Ø
6868         move.b  6(aØ,d1.w),dØ      ; dØ := Byte aus Puffer
686C         move.w  d1,2(aØ)           ; BufferTail setzen
687Ø         rts
6872
6872 FE6872  moveq   #-1,dØ             ; Flag 'Puffer leer'
6874         rts
6876
6876 ;--------------------------------   Wort aus Puffer lesen
6876
6876 FE6876  move.w  2(aØ),d1           ; d1 := BufferTail
687A         cmp.w   (aØ),d1            ; = BufferHead?
687C         beq.s   FE6872             ; ja: Puffer leer ->
687E         addq.w  #2,d1              ; + 2
688Ø         cmp.w   4(aØ),d1           ; < Pufferlänge?
6884         blt.s   FE6888             ; ja: ->
6886         clr.w   d1                 ; d1 := Ø
6888 FE6888  moveq   #Ø,dØ              ; dØ := Ø
688A         move.w  6(aØ,d1.w),dØ      ; dØ := Wort aus Puffer
688E         move.w  d1,2(aØ)           ; BufferTail setzen
6892         rts
6894
6894 ;--------------------------------   Freien Platz im Puffer bestimmen
6894
6894 ; Diese Routine sollte eigentlich die Länge des Pufferinhalts bestimmen,
6894 ; wie sich eindeutig aus dem Aufruf bei FE6FAA ergibt. Tatsächlich
6894 ; ergibt sie aber Ø, wenn der Puffer leer ist, sonst die Zahl der
6894 ; freien Bytes im Puffer. Bei allen anderen Aufrufen hat dieser Fehler
6894 ; keinen Einfluß, weil nur auf Ø abgefragt wird.
6894
6894 FE6894  moveq   #Ø,dØ              ; dØ := Ø
```

```
6896          move.w  2(aØ),dØ            ; dØ := BufferTail
689A          sub.w   (aØ),dØ             ; - BufferHead
689C          bpl.s   FE68A2              ; Ergebnis positiv: ->
689E          add.w   4(aØ),dØ            ; Pufferlänge addieren
68A2 FE68A2   rts
68A4
68A4 ;--------------------------------   cd_Clear
68A4
68A4 FE68A4   movea.l 18(a1),aØ           ; aØ := io_Unit -> ConUnit
68A8          move.w  128(aØ),12A(aØ)     ; BufferTail := BufferHead
68AE          rts
68BØ
68BØ ;--------------------------------   Zeile ab Kursor löschen
68BØ
68BØ FE68BØ   movem.l d2-d3,-(a7)         ; Register retten
68B4          bsr     FE66D8              ; ---> Lock Display
68B8          movea.l 22E(a2),a1          ; a1 -> wd_RPort
68BC          move.w  24(a1),dØ           ; dØ := rp_cp_x
68CØ          ext.l   dØ                  ; dØ auf Langwort erweitern
68C2          move.w  26(a1),d1           ; d1 := rp_cp_y
68C6          sub.w   3E(a1),d1           ; - rp_TxBaseLine
68CA          ext.l   d1                  ; d1 auf Langwort erweitern
68CC          moveq   #Ø,d2               ; d2 := Ø
68CE          move.w  36(a2),d2           ; d2 := cu_XRExtant
68D2          move.l  d1,d3               ; d3 := d1
68D4          movea.l 34(a1),aØ           ; aØ := rp_Font
68D8          add.w   14(aØ),d3           ; d3 := d3 + tf_YSize (Fonthöhe)
68DC          subq.w  #1,d3               ; - 1
68DE          bsr     FE693Ø              ; ---> Rechteck im Display löschen
68E2          bsr     FE678Ø              ; ---> Unlock Display
68E6          movem.l (a7)+,d2-d3         ; Register wiederherstellen
68EA          rts
68EC
68EC ;--------------------------------   Display ab Kursor löschen
68EC
68EC FE68EC   movem.l d2-d3,-(a7)         ; Register retten
68FØ          bsr     FE66D8              ; ---> Lock Display
68F4          bsr.s   FE68BØ              ; ---> Zeile ab Kursor löschen
68F6          movea.l 22E(a2),a1          ; a1 := wd_RPort
68FA          moveq   #Ø,dØ               ; dØ := Ø
```

```
68FC          move.w  32(a2),d0          ; d0 := cu_XROrigin = xmin
6900          move.w  26(a1),d1          ; d1 := rp_cp_y
6904          sub.w   3E(a1),d1          ; - rp_TxBaseLine
6908          movea.l 34(a1),a0          ; a0 := rp_Font
690C          add.w   14(a0),d1          ; d1 := d1 + tf_YSize = ymin
6910          ext.l   d1                 ; d1 auf Langwort erweitern
6912          moveq   #0,d2              ; d2 := 0
6914          move.w  36(a2),d2          ; d2 := cu_XRExtant = xmax
6918          moveq   #0,d3              ; d3 := 0
691A          move.w  38(a2),d3          ; d3 := cu_YRExtant = ymax
691E          cmp.l   d1,d3              ; d3 < d1?
6920          blt.s   FE6926             ; ja: ->
6922          bsr     FE6930             ; ---> Rechteck im Display löschen
6926 FE6926   bsr     FE6780             ; ---> Unlock Display
692A          movem.l (a7)+,d2-d3        ; Register wiederherstellen
692E          rts
6930
6930 ;--------------------------------   Rechteck im Display löschen
6930
6930 FE6930   movem.l d4-d7/a2,-(a7)     ; Register retten
6934          movea.l a1,a2              ; a2 := a1 = wd_RPort
6936          movem.l 28(a1),d4-d5       ; d4, d5 := Minterms
693C          move.l  8(a1),d6           ; d6 := rp_AreaPtrn
6940          move.b  19(a1),d7          ; d7 := rp_FgPen
6944          movem.l d0-d1,-(a7)        ; Register retten
6948          moveq   #0,d0              ; d0 := 0
694A          cmpi.b  #1,1C(a1)          ; rp_DrawMode = JAM2?
6950          bne.s   FE6956             ; nein: ->
6952          move.b  1A(a1),d0          ; d0 := rp_BgPen
6956 FE6956   move.l  a6,-(a7)           ; a6 retten
6958          movea.l 34(a6),a6          ; a6 := GfxBase
695C          jsr     -156(a6)           ; ---> SetAPen
6960          movea.l (a7)+,a6           ; a6 wiederherstellen
6962          movem.l (a7)+,d0-d1        ; Register wiederherstellen
6966          movea.l a2,a1              ; a1 := a2 := wd_RPort
6968          ext.l   d0                 ; xmin,
696A          ext.l   d1                 ; ymin,
696C          ext.l   d2                 ; xmax und
696E          ext.l   d3                 ; ymax auf Langwort erweitern
6970          move.l  a6,-(a7)           ; a6 retten
```

```
6972          movea.l 34(a6),a6           ; a6 := GfxBase
6976          jsr     -132(a6)            ; ---> RectFill
697A          movea.l (a7)+,a6            ; a6 wiederherstellen
697C          movea.l a2,a1               ; a1 := a2 = wd_RPort
697E          movem.l d4-d5,28(a1)        ; rp_Minterms,
6984          move.l  d6,8(a1)            ; rp_AreaPtrn und
6988          move.b  d7,19(a1)           ; rp_FgPen wiederherstellen
698C          movem.l (a7)+,d4-d7/a2      ; Register wiederherstellen
6990          rts
6992
6992          DC.W    0
6994
6994 ;-------------------------------   Kursormuster initialisieren
6994
6994 FE6994   clr.l   108(a2)             ; cu_AreaPtrn := 0
6998          clr.b   107(a2)             ; cu_AreaPtSz := 0
699C          moveq   #0,d0               ; d0 := 0
699E FE699E   rts
69A0
69A0 ;-------------------------------   Kursor einschalten
69A0
69A0 FE69A0   bset.b  #1,2A4(a2)          ; Kursorschalter ein
69A6          bne.s   FE699E              ; war schon ein: fertig ->
69A8          tst.b   2A2(a2)             ; Kursor gesperrt?
69AC          bpl.s   FE699E              ; ja: Fertig ->
69AE
69AE ;-------------------------------   Kursor umschalten
69AE
69AE FE69AE   bsr     FE66D8              ; ---> Lock Display
69B2          movea.l 22E(a2),a1          ; a1 := wd_RPort
69B6          move.b  1C(a1),-(a7)        ; rp_DrawMode auf Stack legen
69BA          moveq   #2,d0               ; d0 := 2 = rp_Complement
69BC          move.l  a6,-(a7)            ; a6 retten
69BE          movea.l 34(a6),a6           ; a6 := GfxBase
69C2          jsr     -162(a6)            ; ---> SetDrMode
69C6          movea.l (a7)+,a6            ; a6 wiederherstellen
69C8          movea.l 22E(a2),a1          ; a1 := wd_RPort
69CC          movem.l d2-d3,-(a7)         ; Register retten
69D0          move.w  3E(a2),d0           ; d0 := cu_XCCP
69D4          mulu    2E(a2),d0           ; * cu_XRSize
```

```
69D8          add.w   32(a2),d0           ; + cu_XROrigin = xmin
69DC          move.w  40(a2),d1           ; d1 := cu_YCCP
69E0          mulu    30(a2),d1           ; * cu_YRSize
69E4          add.w   34(a2),d1           ; + cu_YROrigin = ymin
69E8          move.l  d0,d2               ; d2 := d0 = xmin
69EA          add.w   2E(a2),d2           ; + cu_XRSize
69EE          subq.w  #1,d2               ; - 1 = xmax
69F0          move.l  d1,d3               ; d3 := ymin
69F2          add.w   30(a2),d3           ; + cu_YRSize
69F6          subq.w  #1,d3               ; - 1 = ymax
69F8          move.l  a6,-(a7)            ; a6 retten
69FA          movea.l 34(a6),a6           ; a6 := GfxBase
69FE          jsr     -132(a6)            ; ---> RectFill
6A02          movea.l (a7)+,a6            ; a6 wiederherstellen
6A04          movem.l (a7)+,d2-d3         ; Register wiederherstellen
6A08          movea.l 22E(a2),a1          ; a1 := wd_RPort
6A0C          moveq   #0,d0               ; d0 := 0
6A0E          move.b  (a7)+,d0            ; d0 := rp_DrawMode vom Stack
6A10          move.l  a6,-(a7)            ; a6 retten
6A12          movea.l 34(a6),a6           ; a6 := GfxBase
6A16          jsr     -162(a6)            ; ---> SetDrMode
6A1A          movea.l (a7)+,a6            ; a6 wiederherstellen
6A1C          bra     FE6780              ; ---> Unlock Display
6A20
6A20 ;--------------------------------   Kursor ausschalten
6A20
6A20 FE6A20   bclr.b  #1,2A4(a2)          ; Kursorschalter aus
6A26          beq.s   FE6A4C              ; war schon aus: ->
6A28          tst.b   2A2(a2)             ; Kursor gesperrt?
6A2C          bpl.s   FE6A4C              ; ja: Fertig ->
6A2E          bra     FE69AE              ; ---> Kursor umschalten
6A32
6A32 ;--------------------------------   Kursor löschen
6A32
6A32 FE6A32   movem.l d0-d1/a0-a1,-(a7)   ; Register retten
6A36          addq.b  #1,2A2(a2)          ; Kursorsperre erhöhen
6A3A          bgt.s   FE6A48              ; war schon gesperrt: Fertig ->
6A3C          btst.b  #1,2A4(a2)          ; Kursorschalter aus?
6A42          beq.s   FE6A48              ; ja: ->
6A44          bsr     FE69AE              ; ---> Kursor umschalten
```

```
6A48 FE6A48  movem.l (a7)+,dØ-d1/aØ-a1    ; Register wiederherstellen
6A4C FE6A4C  rts
6A4E
6A4E ;-------------------------------   Kursor setzen
6A4E
6A4E FE6A4E  movem.l dØ-d1/aØ-a1,-(a7)    ; Register retten
6A52         subq.b  #1,2A2(a2)           ; Kursorsperre erniedrigen
6A56         bpl.s   FE6A48               ; noch gesperrt: Fertig ->
6A58         btst.b  #1,2A4(a2)           ; Kursorschalter aus?
6A5E         beq.s   FE6A48               ; ja: ->
6A6Ø         bsr     FE69AE               ; ---> Kursor umschalten
6A64         bra.s   FE6A48               ; ---> Abschluß
6A66
6A66 ;-------------------------------   Kursor umsetzen
6A66
6A66 FE6A66  moveq   #Ø,dØ                ; dØ := Ø
6A68         move.l  26(a2),d1            ; d1 := cu_XCP (Zeichenposition)
6A6C         cmp.l   3E(a2),d1            ; = cu_XCCP (Kursorposition)?
6A7Ø         beq.s   FE6A8Ø               ; ja: Fertig ->
6A72         bclr.b  #Ø,2A4(a2)           ; Zeilenverkettungsflag löschen
6A78         bsr.s   FE6A32               ; ---> Kursor löschen
6A7A         move.l  d1,3E(a2)            ; cu_XCCP := d1
6A7E         bsr.s   FE6A4E               ; ---> Kursor setzen
6A8Ø FE6A8Ø  rts
6A82
6A82 ;-------------------------------   Kursor dØ Zeilen aufwärts
6A82
6A82 FE6A82  move.w  28(a2),d1            ; d1 := cu_YCP
6A86         sub.w   dØ,d1                ; - dØ
6A88         bcs.s   FE6A9Ø               ; < Ø: ->
6A8A         move.w  d1,28(a2)            ; als cu_YCP speichern
6A8E         bra.s   FE6A66               ; ---> Kursor umsetzen
6A9Ø
6A9Ø FE6A9Ø  clr.w   28(a2)               ; cu_YCP := Ø
6A94         btst.b  #5,124(a2)           ; AutoScroll-Flag gesetzt?
6A9A         beq.s   FE6AA4               ; nein: ->
6A9C         moveq   #Ø,dØ                ; dØ := Ø = Dx
6A9E FE6A9E  ext.l   d1                   ; d1 auf Langwort erweitern = Dy
6AAØ         bsr     FE6B3E               ; ---> Scroll Display
6AA4 FE6AA4  bra.s   FE6A66               ; ---> Kursor umsetzen
```

```
6AA6
6AA6 ;--------------------------------   Kursor dØ Zeilen abwärts
6AA6
6AA6 FE6AA6  move.w  28(a2),d1           ; d1 := cu_YCP
6AAA          add.w   dØ,d1              ; + dØ
6AAC          cmp.w   2C(a2),d1          ; > cu_YMax?
6ABØ          bhi.s   FE6AB8             ; ja: ->
6AB2          move.w  d1,28(a2)          ; als cu_YCP speichern
6AB6          bra.s   FE6A66             ; ---> Kursor umsetzen
6AB8
6AB8 FE6AB8  move.w  2C(a2),28(a2)       ; cu_YCP := cu_YMax
6ABE          btst.b  #5,124(a2)         ; AutoScroll-Flag gesetzt?
6AC4          beq.s   FE6AA4             ; nein: ->
6AC6          moveq   #Ø,dØ              ; dØ := Ø
6AC8          sub.w   2C(a2),d1          ; d1 := d1 - cu_YMax
6ACC          bra.s   FE6A9E             ; ---> Scroll, Kursor umsetzen
6ACE
6ACE ;--------------------------------   Kursor dØ Spalten vor
6ACE
6ACE FE6ACE  add.w   26(a2),dØ           ; dØ := dØ + cu_XCP
6AD2          cmp.w   2A(a2),dØ          ; > cu_XMax?
6AD6          bhi.s   FE6ADE             ; ja: ->
6AD8          move.w  dØ,26(a2)          ; als cu_XCP speichern
6ADC FE6ADC  bra.s   FE6A66             ; ---> Kursor umsetzen
6ADE
6ADE FE6ADE  btst.b  #6,124(a2)          ; AutoWrap-Flag gesetzt?
6AE4          bne.s   FE6AEE             ; ja: ->
6AE6          move.w  2A(a2),26(a2)      ; cu_XCP := cu_XMax
6AEC          bra.s   FE6ADC             ; ---> Kursor umsetzen
6AEE
6AEE FE6AEE  clr.w   26(a2)              ; cu_XCP := Ø
6AF2          sub.w   2A(a2),dØ          ; dØ := dØ - cu_XMax
6AF6          subq.w  #1,dØ              ; - 1
6AF8          move.w  dØ,-(a7)           ; auf Stack legen
6AFA          moveq   #1,dØ              ; dØ := 1
6AFC          bsr.s   FE6AA6             ; ---> Kursor dØ Zeilen abwärts
6AFE          move.w  (a7)+,dØ           ; dØ wiederherstellen
6BØØ          bne.s   FE6ACE             ; nicht Ø: Kursor dØ Zeilen vor ->
6BØ2          bset.b  #Ø,2A4(a2)         ; Zeilenverkettungsflag setzen
6BØ8          rts
```

```
6BØA
6BØA ;--------------------------------   Kursor dØ Spalten zurück
6BØA
6BØA FE6BØA  move.w  26(a2),d1           ; d1 := cu_XCP
6BØE         sub.w   dØ,d1               ; - dØ
6B1Ø         bcs.s   FE6B1A              ; < Ø: ->
6B12         move.w  d1,26(a2)           ; als cu_XCP speichern
6B16 FE6B16  bra     FE6A66              ; ---> Kursor umsetzen
6B1A
6B1A FE6B1A  btst.b  #6,124(a2)          ; AutoWrap-Flag gesetzt?
6B2Ø         bne.s   FE6B28              ; ja: ->
6B22         clr.w   26(a2)              ; cu_XCP := Ø
6B26         bra.s   FE6B16              ; ---> Kursor setzen
6B28
6B28 FE6B28  move.w  2A(a2),26(a2)       ; cu_XCP := cu_XMax
6B2E         neg.w   d1                  ; d1 := -d1
6B3Ø         subq.w  #1,d1               ; - 1
6B32         move.w  d1,-(a7)            ; auf Stack legen
6B34         moveq   #1,dØ               ; dØ := 1
6B36         bsr     FE6A82              ; ---> Kursor dØ Zeilen aufwärts
6B3A         move.w  (a7)+,dØ            ; dØ wiederherstellen
6B3C         bra.s   FE6BØA              ; ---> Kursor dØ Spalten zurück
6B3E
6B3E ;--------------------------------   Scroll Display
6B3E
6B3E FE6B3E  movem.l d2-d5,-(a7)         ; Register retten
6B42         bsr     FE66D8              ; ---> Lock Display
6B46         movea.l 22E(a2),a1          ; a1 := wd_RPort
6B4A         muls    2E(a2),dØ           ; dØ := dØ * cu_XRSize = dx
6B4E         muls    3Ø(a2),d1           ; d1 := d1 * cu_YRSize = dy
6B52         move.w  32(a2),d2           ; d2 := cu_XROrigin = xmin
6B56         ext.l   d2                  ; d2 auf Langwort erweitern
6B58         move.w  34(a2),d3           ; d3 := cu_YROrigin = ymin
6B5C         ext.l   d3                  ; d3 auf Langwort erweitern
6B5E         move.w  36(a2),d4           ; d4 := cu_XRExtant = xmax
6B62         ext.l   d4                  ; d4 auf Langwort erweitern
6B64         move.w  38(a2),d5           ; d5 := cu_YRExtant
6B68         ext.l   d5                  ; d5 auf Langwort erweitern
6B6A         bsr     FE6A32              ; ---> Kursor löschen
6B6E         move.l  a6,-(a7)            ; a6 retten
```

```
6B7Ø          movea.l 34(a6),a6           ; a6 := GfxBase
6B74          jsr     -18C(a6)            ; ---> ScrollRaster
6B78          movea.l (a7)+,a6            ; a6 wiederherstellen
6B7A          bsr     FE6A4E              ; ---> Kursor setzen
6B7E          bsr     FE678Ø              ; ---> Unlock Display
6B82          movem.l (a7)+,d2-d5         ; Register wiederherstellen
6B86          rts
6B88
6B88 ;--------------------------------   Initialisierung console.device
6B88
6B88 FE6B88  movem.l aØ/a2,-(a7)          ; Register retten
6B8C          lea     14C(pc),aØ          ; aØ := FE6CDA -> Vektortabelle
6B9Ø          lea     FC(pc),a1           ; a1 := FE6C8E -> Tabelle f InitStruct
6B94          lea     C(pc),a2            ; a2 := FE6BA2 -> Init-Routine
6B98          move.l  #B8,dØ              ; dØ := 184 Bytes Speicherbedarf
6B9E          bra     FE52A4              ; ---> Device einrichten
6BA2
6BA2 ;------ Init-Routine für MakeLibrary
6BA2
6BA2 FE6BA2  move.l  a6,-(a7)            ; a6 retten
6BA4          movea.l dØ,aØ               ; aØ := dØ = LibBase
6BA6          move.l  a6,26(aØ)           ; SysBase speichern
6BAA          movea.l dØ,a6               ; a6 := dØ = LibBase
6BAC          lea     181(pc),a1          ; a1 := FE6D2F -> 'graphics.library'
6BBØ          moveq   #Ø,dØ               ; Version beliebig
6BB2          move.l  a6,-(a7)            ; a6 retten
6BB4          movea.l 26(a6),a6           ; a6 := SysBase
6BB8          jsr     -228(a6)            ; ---> OpenLibrary
6BBC          movea.l (a7)+,a6            ; a6 wiederherstellen
6BBE          move.l  dØ,34(a6)           ; GfxBase speichern
6BC2          beq     FE6C8A              ; Nicht geöffnet: ->
6BC6          lea     185(pc),a1          ; a1 := FE6D4D -> 'keymap.resource'
6BCA          move.l  a6,-(a7)            ; a6 retten
6BCC          movea.l 26(a6),a6           ; a6 := SysBase
6BDØ          jsr     -1F2(a6)            ; ---> OpenResource
6BD4          movea.l (a7)+,a6            ; a6 wiederherstellen
6BD6          tst.l   dØ                  ; Resource geöffnet?
6BD8          beq     FE6C7A              ; Nein: ->
6BDC          movea.l dØ,aØ               ; aØ := dØ -> keymap.resource
6BDE          lea     E(aØ),aØ            ; aØ -> KeyMap List Header
```

```
6BE2           move.l   (aØ),dØ            ; dØ := lh_Head
6BE4           beq      FE6C7A             ; Liste leer: ->
6BE8           movea.l  dØ,aØ              ; aØ -> KeyMap Node
6BEA           lea      E(aØ),aØ           ; aØ -> KeyMap
6BEE           lea      98(a6),a1          ; a1 -> Zielbereich für KeyMap
6BF2           moveq    #7,dØ              ; Zähler setzen: 8 Adressen
6BF4 FE6BF4    move.l   (aØ)+,(a1)+        ; KeyMap-Adressen kopieren
6BF6           dbra     dØ,FE6BF4          ; 7mal wiederholen ->
6BFA           lea      3C(a6),aØ          ; aØ -> ConUnits List Header
6BFE           move.l   aØ,(aØ)            ; Macro NewList
6CØØ           addq.l   #4,(aØ)            ; :
6CØ2           clr.l    4(aØ)              ; :
6CØ6           move.l   aØ,8(aØ)           ; :
6CØA           lea      134(pc),aØ         ; aØ := FE6D4Ø -> 'input.device'
6CØE           moveq    #Ø,dØ              ; Version beliebig
6C1Ø           lea      52(a6),a1          ; a1 -> IORequest
6C14           moveq    #Ø,d1              ; d1 := Ø = Flags
6C16           move.l   a6,-(a7)           ; a6 retten
6C18           movea.l  26(a6),a6          ; a6 := SysBase
6C1C           jsr      -1BC(a6)           ; ---> OpenDevice
6C2Ø           movea.l  (a7)+,a6           ; a6 wiederherstellen
6C22           tst.l    dØ                 ; Device geöffnet?
6C24           beq.s    FE6C2C             ; ja: ->
6C26           clr.l    66(a6)             ; io_Device löschen
6C2A           bra.s    FE6C74             ; --->
6C2C
6C2C FE6C2C    movea.l  a6,a1              ; a1 := a6 = ConsoleBase
6C2E           adda.w   #-2A,a1            ; a1 -> 'jmp CDInputHandler'
6C32           move.l   a1,94(a6)          ; is_Code := a1
6C36           move.l   a6,9Ø(a6)          ; is_Data := a6
6C3A           suba.w   #22,a7             ; MsgPort im Stack einrichten
6C3E           movea.l  a7,aØ              ; aØ := a7 -> MsgPort
6C4Ø           bsr      FE538C             ; ---> Signal zuweisen
6C44           tst.l    dØ                 ; Signal zugewiesen?
6C46           bmi.s    FE6C7Ø             ; nein: ->
6C48           move.l   a7,6Ø(a6)          ; mn_ReplyPort in IORequest eintragen
6C4C           move.w   #9,6E(a6)          ; io_Command := 'ind_AddHandler'
6C52           lea      82(a6),a1          ; a1 -> InputHandler-Struktur
6C56           move.l   a1,7A(a6)          ; io_Data -> InputHandler-Struktur
6C5A           lea      52(a6),a1          ; a1 -> IORequest
```

```
6C5E          move.l  a6,-(a7)          ; a6 retten
6C60          movea.l 26(a6),a6         ; a6 := SysBase
6C64          jsr     -1C8(a6)          ; ---> DoIO
6C68          movea.l (a7)+,a6          ; a6 wiederherstellen
6C6A          movea.l a7,a0             ; a0 := a7 -> MsgPort
6C6C          bsr     FE53D2            ; ---> Signal freigeben
6C70 FE6C70   adda.w  #22,a7            ; MsgPort beseitigen
6C74 FE6C74   move.l  a6,d0             ; d0 := a6 = ConsoleDevBase
6C76 FE6C76   movea.l (a7)+,a6          ; a6 wiederherstellen
6C78          rts
6C7A
6C7A FE6C7A   movea.l 34(a6),a1         ; a1 := GfxBase
6C7E          move.l  a6,-(a7)          ; a6 retten
6C80          movea.l 26(a6),a6         ; a6 := SysBase
6C84          jsr     -19E(a6)          ; ---> CloseLibrary
6C88          movea.l (a7)+,a6          ; a6 wiederherstellen
6C8A FE6C8A   moveq   #0,d0             ; d0 := 0
6C8C          bra.s   FE6C76            ; ---> Abschluß
6C8E
6C8E ;--------------------------------   Tabelle für InitStruct
6C8E
6C8E FE6C8E   DC.W    E000              ; InitByte
6C90          DC.W    8                 ; ln_Type
6C92          DC.B    3                 ; = nt_Device
6C93          DC.B    0
6C94          DC.W    C000              ; InitLong
6C96          DC.W    A                 ; lib_Name
6C98          DC.L    FE5128            ; -> 'console.device'
6C9C          DC.W    E000              ; InitByte
6C9E          DC.W    E                 ; lib_Flags
6CA0          DC.B    6                 ; = Changed,Sumused
6CA1          DC.B    0
6CA2          DC.W    D000              ; InitWord
6CA4          DC.W    14                ; Version
6CA6          DC.W    21                ; = 33
6CA8          DC.W    D000              ; InitWord
6CAA          DC.W    16                ; Revision
6CAC          DC.W    67                ; = 103
6CAE          DC.W    C000              ; InitLong
6CB0          DC.W    2A
```

```
6CB2          DC.L    FE6CEE              ; -> Kommando-Tabelle
6CB6          DC.W    C000                ; InitLong
6CB8          DC.W    2E
6CBA          DC.L    FE6D22              ; -> Flag-Tabelle
6CBE          DC.W    D000                ; InitWord
6CC0          DC.W    32
6CC2          DC.W    D                   ; Letztes Kommando + 1
6CC4          DC.W    E000                ; InitByte
6CC6          DC.W    8A                  ; ln_Type
6CC8          DC.B    2                   ; = nt_Interrupt
6CC9          DC.B    0
6CCA          DC.W    E000                ; InitByte
6CCC          DC.W    8B                  ; ln_Pri
6CCE          DC.B    0                   ; = 0
6CCF          DC.B    0
6CD0          DC.W    C000                ; InitLong
6CD2          DC.W    8C                  ; ln_Name in Interrupt-Struktur
6CD4          DC.L    FE5128              ; -> 'console.device'
6CD8          DC.W    0
6CDA
6CDA ;--------------------------------   Vektoren für Sprungliste
6CDA
6CDA FE6CDA   DC.W    -1
6CDC          DC.W    036A                ; FE7044  Open
6CDE          DC.W    0494                ; FE716E  Close
6CE0          DC.W    0088                ; FE6D62  Expunge
6CE2          DC.W    E61A                ; FE52F4  Extfunct
6CE4          DC.W    E482                ; FE515C  BeginIO
6CE6          DC.W    E546                ; FE5220  AbortIO
6CE8          DC.W    0112                ; FE6DEC  CDInputHandler
6CEA          DC.W    04E2                ; FE71BC  RawKeyConvert
6CEC          DC.W    -1
6CEE
6CEE ;--------------------------------   Adressentabelle für IO-Kommandos
6CEE
6CEE FE6CEE   DC.L    FE5314              ; $0 cd_Invalid
6CF2          DC.L    FE531C              ; $1 cd_Reset
6CF6          DC.L    FE73C0              ; $2 cd_Read
6CFA          DC.L    FE780C              ; $3 cd_Write
6CFE          DC.L    FE5388              ; $4 cd_Update
```

```
6DØ2          DC.L    FE68A4              ; $5 cd_Clear
6DØ6          DC.L    FE532A              ; $6 cd_Stop
6DØA          DC.L    FE533A              ; $7 cd_Start
6DØE          DC.L    FE5366              ; $8 cd_Flush
6D12          DC.L    FE6FD4              ; $9 cd_AskKeyMap
6D16          DC.L    FE6FE4              ; $A cd_SetKeyMap
6D1A          DC.L    FE6FF4              ; $B cd_AskDefaultKeyMap
6D1E          DC.L    FE7ØØØ              ; $C cd_SetDefaultKeyMap
6D22
6D22 ;--------------------------------   Kommando-Flags
6D22
6D22 FE6D22   DC.B    -1,-1,Ø,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
6D2F
6D2F ;--------------------------------   Namen
6D2F
6D2F FE6D2F   DC.B    'graphics.library',Ø
6D4Ø
6D4Ø FE6D4Ø   DC.B    'input.device',Ø
6D4D
6D4D FE6D4D   DC.B    'keymap.resource',Ø
6D5D
6D5D          DC.B    'usa',Ø,Ø
6D62
6D62 ;--------------------------------   cd_Expunge
6D62
6D62 FE6D62   tst.w   2Ø(a6)              ; Open-Zähler = Ø?
6D66          bne     FE52EC              ; nein: DelExp-Flag setzen
6D6A          tst.l   66(a6)              ; io_Device = Ø?
6D6E          beq.s   FE6DBA              ; ja: ->
6D7Ø          suba.w  #22,a7              ; MsgPort im Stack einrichten
6D74          movea.l a7,aØ               ; aØ -> MsgPort
6D76          bsr     FE538C              ; ---> Signal zuweisen
6D7A          tst.l   dØ                  ; Signal zugewiesen?
6D7C          bmi.s   FE6DE2              ; nein: ->
6D7E          move.l  a7,6Ø(a6)           ; MsgPort als ReplyPort eintragen
6D82          move.w  #A,6E(a6)           ; io_Command := 'cd_SetKeyMap'
6D88          lea     82(a6),a1           ; a1 -> KeyMap
6D8C          move.l  a1,7A(a6)           ; io_Data := a1
6D9Ø          lea     52(a6),a1           ; a1 -> IORequest
6D94          move.l  a6,-(a7)            ; a6 retten
```

```
6D96          movea.l 26(a6),a6           ; a6 := SysBase
6D9A          jsr     -1C8(a6)            ; ---> DoIO
6D9E          movea.l (a7)+,a6            ; a6 wiederherstellen
6DA0          movea.l a7,a0               ; a0 -> MsgPort
6DA2          bsr     FE53D2              ; ---> Signal freigeben
6DA6          adda.w  #22,a7              ; MsgPort im Stack beseitigen
6DAA          lea     52(a6),a1           ; a1 -> IORequest
6DAE          move.l  a6,-(a7)            ; a6 retten
6DB0          movea.l 26(a6),a6           ; a6 := SysBase
6DB4          jsr     -1C2(a6)            ; ---> CloseDevice
6DB8          movea.l (a7)+,a6            ; a6 wiederherstellen
6DBA FE6DBA   move.l  38(a6),d0           ; d0 := IntuiBase
6DBE          beq.s   FE6DCE              ; Intuition nicht geöffnet: ->
6DC0          movea.l d0,a1               ; a1 := d0 = IntuiBase
6DC2          move.l  a6,-(a7)            ; a6 retten
6DC4          movea.l 26(a6),a6           ; a6 := SysBase
6DC8          jsr     -19E(a6)            ; ---> CloseLibrary
6DCC          movea.l (a7)+,a6            ; a6 wiederherstellen
6DCE FE6DCE   movea.l 34(a6),a1           ; a1 := GfxBase
6DD2          move.l  a6,-(a7)            ; a6 retten
6DD4          movea.l 26(a6),a6           ; a6 := SysBase
6DD8          jsr     -19E(a6)            ; ---> CloseLibrary
6DDC          movea.l (a7)+,a6            ; a6 wiederherstellen
6DDE          bra     FE52BC              ; ---> Device entfernen
6DE2
6DE2 FE6DE2   adda.w  22,a7               ; '#' fehlt: Guru kommt sofort!!!
6DE6          bra     FE52EC              ; ---> DelExp-Flag setzen
6DEA
6DEA          DC.W    0
6DEC
6DEC ;--------------------------------   CDInputHandler
6DEC
6DEC FE6DEC   movem.l d2/d4/a2-a4/a6,-(a7) ; Register retten
6DF0          link    a5,#-80             ; 128 Bytes Stackrahmen schaffen
6DF4          movea.l a1,a6               ; a6 := a1 -> Console.device
6DF6          move.l  a0,d0               ; d0 := a0 -> Input Event
6DF8          beq     FE6E8A              ; Kein Event: fertig ->
6DFC          moveq   #0,d2               ; d2 := 0
6DFE FE6DFE   movea.l d0,a3               ; a3 -> Event
6E00          cmpi.b  #3,4(a3)            ; ieClass_Event?
```

```
6EØ6          bne.s   FE6E34          ; nein: ->
6EØ8          cmpi.w  #1,6(a3)        ; ieCode_NewActive?
6EØE          bne.s   FE6E34          ; nein: ->
6E1Ø          clr.l   4A(a6)          ; Active ConUnit löschen
6E14          movea.l A(a3),a1        ; a1 -> ie_EventAddress
6E18          move.l  a1,4E(a6)       ; als Aktives Window speichern
6E1C          movea.l 3C(a6),a2       ; a2 := lh_Head (ConUnits Liste)
6E2Ø FE6E2Ø   tst.l   (a2)            ; Ende der Liste?
6E22          beq     FE6E64          ; ja: ->
6E26          cmpa.l  22(a2),a1       ; cu_Window = ie_EventAddress?
6E2A          beq.s   FE6E3Ø          ; ja: ->
6E2C          movea.l (a2),a2         ; a2 -> Nächster Node
6E2E          bra.s   FE6E2Ø          ; ---> Loop
6E3Ø
6E3Ø FE6E3Ø   move.l  a2,4A(a6)       ; Active ConUnit speichern
6E34 FE6E34   btst.b  #B,8(a3)        ; ieQualifier_MultiBroadcast?
6E3A          beq.s   FE6E4C          ; nein: ->
6E3C          movea.l 3C(a6),a2       ; a2 := lh_Head (ConUnits Liste)
6E4Ø FE6E4Ø   tst.l   (a2)            ; Ende der Liste?
6E42          beq.s   FE6E64          ; ja: ->
6E44          bsr.s   FE6EBA          ; ---> Input Event Report erzeugen
6E46          asr.w   #1,d2           ; Flagbit verschieben
6E48          movea.l (a2),a2         ; a2 -> Nächster Node
6E4A          bra.s   FE6E4Ø          ; ---> Loop
6E4C
6E4C FE6E4C   move.l  4A(a6),dØ       ; dØ -> Actual ConUnit
6E5Ø          beq     FE6E64          ; nicht vorhanden: ->
6E54          movea.l dØ,a2           ; a2 -> Actual ConUnit
6E56          cmpi.b  #C,4(a3)        ; ieClass_SizeWindow?
6E5C          bne.s   FE6E62          ; nein: ->
6E5E          bsr     FE7688          ; ---> Ausgaberest erfassen
6E62 FE6E62   bsr.s   FE6EBA          ; ---> Input Event Report erzeugen
6E64 FE6E64   move.l  (a3),dØ         ; dØ := ie_NextEvent
6E66          bne.s   FE6DFE          ; NextEvent vorhanden: Loop ->
6E68          tst.w   d2              ; Event Report erzeugt?
6E6A          beq.s   FE6E8A          ; nein: fertig ->
6E6C          btst.l  #E,d2           ; ie_MultiBroadcast?
6E7Ø          beq.s   FE6E8Ø          ; nein: ->
6E72          movea.l 3C(a6),a2       ; a2 := lh_Head (ConUnits Liste)
6E76 FE6E76   tst.l   (a2)            ; Ende der Liste?
```

```
6E78          beq.s   FE6E8A                ; ja: ->
6E7A          bsr.s   FE6E94                ; ---> ReadRequest bearbeiten
6E7C          movea.l (a2),a2               ; a2 -> Nächster Node
6E7E          bra.s   FE6E76                ; ---> Loop
6E80
6E80 FE6E80   move.l  4A(a6),d0             ; d0 := Actual ConUnit
6E84          beq.s   FE6E8A                ; nicht vorhanden: ->
6E86          movea.l d0,a2                 ; a2 -> ConUnit
6E88          bsr.s   FE6E94                ; ---> ReadRequest bearbeiten
6E8A FE6E8A   moveq   #0,d0                 ; d0 := 0
6E8C          unlk    a5                    ; Stackrahmen freigeben
6E8E          movem.l (a7)+,d2/d4/a2-a4/a6 ; Register wiederherstellen
6E92          rts
6E94
6E94 ;--------------------------------   ReadRequest bearbeiten
6E94
6E94 FE6E94   move.l  a6,-(a7)              ; a6 retten
6E96          movea.l 26(a6),a6             ; a6 := SysBase
6E9A          jsr     -84(a6)               ; ---> Forbid
6E9E          movea.l (a7)+,a6              ; a6 wiederherstellen
6EA0          movea.l 14(a2),a1             ; a1 := lh_Head (mp_MsgList)
6EA4          tst.l   (a1)                  ; Ende der Liste?
6EA6          beq.s   FE6EAC                ; ja: ->
6EA8          bsr     FE73C0                ; ---> cd_Read
6EAC FE6EAC   move.l  a6,-(a7)              ; a6 retten
6EAE          movea.l 26(a6),a6             ; a6 := SysBase
6EB2          jsr     -8A(a6)               ; ---> Permit
6EB6          movea.l (a7)+,a6              ; a6 wiederherstellen
6EB8          rts
6EBA
6EBA ;--------------------------------   InputEvent-Report erzeugen
6EBA
6EBA FE6EBA   moveq   #0,d0                 ; d0 := 0
6EBC          move.b  4(a3),d0              ; d0 := ie_Class
6EC0          cmpi.b  #12,d0                ; > ieClass_Max?
6EC4          bhi     FE6F88                ; ja: ->
6EC8          move.w  d0,d1                 ; d1 := d0 = ie_Class
6ECA          andi.w  #7,d0                 ; d0 := ie_Class mod 8
6ECE          lsr.w   #3,d1                 ; d1 := ie_Class \ 8
6ED0          lea     125(a2),a0            ; a0 -> RawEvents Flags
```

```
6ED4        btst.b  dØ,Ø(aØ,d1.w)       ; Flag für ie_Class gesetzt?
6ED8        beq     FE6F88              ; nein: ->
6EDC        move.l  12(a3),dØ           ; dØ := tv_Micro
6EEØ        bclr.l  #1F,dØ              ; Bit 31 löschen
6EE4        move.l  dØ,-(a7)            ; auf Stack
6EE6        move.l  E(a3),dØ            ; dØ := tv_Secs
6EEA        bclr.l  #1F,dØ              ; Bit 31 löschen
6EEE        move.l  dØ,-(a7)            ; auf Stack
6EFØ        moveq   #Ø,dØ               ; dØ := Ø
6EF2        move.w  C(a3),dØ            ; dØ := ie_Y
6EF6        move.l  dØ,-(a7)            ; auf Stack
6EF8        move.w  A(a3),dØ            ; dØ := ie_X
6EFC        move.l  dØ,-(a7)            ; auf Stack
6EFE        move.w  8(a3),dØ            ; dØ := ie_Qualifier
6FØ2        move.l  dØ,-(a7)            ; auf Stack
6FØ4        move.w  6(a3),dØ            ; dØ := ie_Code
6FØ8        move.l  dØ,-(a7)            ; auf Stack
6FØA        moveq   #Ø,dØ               ; dØ := Ø
6FØC        move.b  5(a3),dØ            ; dØ := ie_SubClass
6F1Ø        move.w  dØ,-(a7)            ; auf Stack
6F12        move.b  4(a3),dØ            ; dØ := ie_Class
6F16        move.w  dØ,-(a7)            ; auf Stack
6F18        movea.l a7,a1               ; a1 -> Ausgabedaten im Stack
6F1A        movem.l a2-a3,-(a7)         ; Register retten
6F1E        lea     3C(pc),aØ           ; aØ := FE6F5C -> Formatstring
6F22        lea     128(a2),a3          ; a3 -> PutChData
6F26        lea     54(pc),a2           ; a2 := FE6F7C -> PutChProc
6F2A        move.l  a6,-(a7)            ; a6 retten
6F2C        movea.l 26(a6),a6           ; a6 := SysBase
6F3Ø        jsr     -84(a6)             ; ---> Forbid
6F34        movea.l (a7)+,a6            ; a6 wiederherstellen
6F36        move.l  a6,-(a7)            ; a6 retten
6F38        movea.l 26(a6),a6           ; a6 := SysBase
6F3C        jsr     -2ØA(a6)            ; ---> RawDoFmt
6F4Ø        movea.l (a7)+,a6            ; a6 wiederherstellen
6F42        move.l  a6,-(a7)            ; a6 retten
6F44        movea.l 26(a6),a6           ; a6 := SysBase
6F48        jsr     -8A(a6)             ; ---> Permit
6F4C        movea.l (a7)+,a6            ; a6 wiederherstellen
6F4E        movem.l (a7)+,a2-a3         ; Register wiederherstellen
```

```
6F52          adda.w  #1C,a7              ; Datenbereich im Stack freigeben
6F56          ori.w   #8000,d2            ; EventReport-Flag setzen
6F5A          rts
6F5C
6F5C FE6F5C   DC.B    9B,'%d;%d;%ld;%ld;%ld;%ld;%ld;%ld',7C,0
6F7C
6F7C ;------ PutChProc für RawDoFmt
6F7C
6F7C FE6F7C   tst.w   d0                  ; Textende?
6F7E          beq.s   FE6F86              ; ja: ->
6F80          movea.l a3,a0               ; a0 := a3 -> Ausgabebereich
6F82          bsr     FE6814              ; ---> (d0) in Ausgabebereich
6F86 FE6F86   rts
6F88
6F88 ;------ Eingabe aufbereiten ("Key cooking")
6F88
6F88 FE6F88   movea.l (a3),a4             ; a4 := ie_NextEvent
6F8A          clr.l   (a3)                ; ie_NextEvent := 0
6F8C          move.l  a2,-(a7)            ; a2 retten
6F8E          movea.l a3,a0               ; a0 := a3 -> InputEvent
6F90          lea     -7F(a5),a1          ; a1 -> Ergebnispuffer
6F94          moveq   #7F,d1              ; d1 := 127
6F96          lea     42(a2),a2           ; a2 -> cu_KeyMapStruct
6F9A          jsr     -30(a6)             ; ---> RawKeyConvert
6F9E          movea.l (a7)+,a2            ; a2 wiederherstellen
6FA0          move.l  a4,(a3)             ; ie_NextEvent wiederherstellen
6FA2          move.w  d0,d4               ; d4 := d0 = Zahl der Kodes im Puffer
6FA4          beq.s   FE6FD0              ; keine Kodes: Fertig ->
6FA6          lea     128(a2),a0          ; a0 -> ConUnit-Puffer
6FAA          bsr     FE6894              ; ---> d0 := Länge des Pufferinhalts
6FAE          sub.w   12C(a2),d0          ; - Maximale Länge
6FB2          neg.w   d0                  ; d0 := freier Platz im InpStr Puffer
6FB4          cmp.w   d0,d4               ; genügend Platz für KeyCodes?
6FB6          bgt.s   FE6FD0              ; nein: ->
6FB8          lea     -7F(a5),a4          ; a4 -> Ergebnispuffer
6FBC          bra.s   FE6FC8              ; --->
6FBE
6FBE FE6FBE   move.b  (a4)+,d0            ; d0 := Kode aus Ergebnispuffer
6FC0          lea     128(a2),a0          ; a0 -> ConUnit-Puffer
6FC4          bsr     FE6814              ; ---> (d0) in ConUnit-Puffer kopieren
```

```
6FC8 FE6FC8  dbra    d4,FE6FBE           ; wiederholen bis d4 = 0 ->
6FCC         ori.w   #8000,d2            ; EventReport-Flag setzen
6FD0 FE6FD0  rts
6FD2
6FD2         DC.W    0
6FD4
6FD4 ;-------------------------------   cd_AskKeyMap
6FD4
6FD4 FE6FD4  move.l  a2,-(a7)            ; a2 retten
6FD6         movea.l 18(a1),a0           ; a0 := io_Unit -> ConUnit
6FDA         lea     42(a0),a0           ; a0 -> cu_KeyMapStruct
6FDE         movea.l 28(a1),a2           ; a2 := io_Data
6FE2         bra.s   FE700A              ; ---> KeyMap kopieren
6FE4
6FE4 ;-------------------------------   cd_SetKeyMap
6FE4
6FE4 FE6FE4  move.l  a2,-(a7)            ; a2 retten
6FE6         movea.l 28(a1),a0           ; a0 := io_Data
6FEA         movea.l 18(a1),a2           ; a2 := io_Unit -> ConUnit
6FEE         lea     42(a2),a2           ; a2 -> cu_KeyMapStruct
6FF2         bra.s   FE700A              ; ---> KeyMap kopieren
6FF4
6FF4 ;-------------------------------   cd_AskDefaultKeyMap
6FF4
6FF4 FE6FF4  move.l  a2,-(a7)            ; a2 retten
6FF6         lea     98(a6),a0           ; a0 -> cd_DefaultKeyMap
6FFA         movea.l 28(a1),a2           ; a2 := io_Data
6FFE         bra.s   FE700A              ; ---> KeyMap kopieren
7000
7000 ;-------------------------------   cd_SetDefaultKeyMap
7000
7000 FE7000  move.l  a2,-(a7)            ; a2 retten
7002         movea.l 28(a1),a0           ; a0 := io_Data
7006         lea     98(a6),a2           ; a2 -> cd_DefaultKeyMap
700A FE700A  moveq   #20,d0              ; d0 := 32 = Länge der KeyMap
700C         cmp.l   24(a1),d0           ; = io_Length?
7010         bne.s   FE703E              ; nein: Fehler ->
7012         moveq   #7,d1               ; d1 := 7 = Kopierzähler
7014         move.l  a6,-(a7)            ; a6 retten
7016         movea.l 26(a6),a6           ; a6 := SysBase
```

```
701A          jsr     -84(a6)             ; ---> Forbid
701E          movea.l (a7)+,a6            ; a6 wiederherstellen
7020 FE7020   move.l  (a0)+,(a2)+         ; KeyMap Adresse kopieren
7022          dbra    d1,FE7020           ; 7 mal wiederholen ->
7026          move.l  a6,-(a7)            ; a6 retten
7028          movea.l 26(a6),a6           ; a6 := SysBase
702C          jsr     -8A(a6)             ; ---> Permit
7030          movea.l (a7)+,a6            ; a6 wiederherstellen
7032          move.l  d0,20(a1)           ; io_Actual := 32
7036          moveq   #0,d0               ; d0 := ok-Kode
7038 FE7038   movea.l (a7)+,a2            ; a2 wiederherstellen
703A          bra     FE5226              ; ---> IORequest beantworten
703E
703E FE703E   moveq   #-4,d0              ; d0 := ioerr_BadLength
7040          bra.s   FE7038              ; ---> Abschluß
7042
7042          DC.W    0
7044
7044 ;--------------------------------    Open
7044
7044 FE7044   bclr.b  #3,E(a6)            ; DelExp-Flag löschen
704A          addq.w  #1,20(a6)           ; Open-Zähler erhöhen
704E          cmpi.l  #-1,d0              ; unit = -1?
7054          bne.s   FE705C              ; nein: ->
7056          move.l  d0,18(a1)           ; io_Unit := -1
705A          rts
705C
705C FE705C   movem.l d2-d3/a2-a4,-(a7)   ; Register retten
7060          movea.l a1,a3               ; a3 := a1 -> IORequest
7062          move.l  28(a3),d0           ; d0 := io_Data
7066          beq     FE711A              ; = 0: Fehler ->
706A          move.l  #2A5,d0             ; d0 := 677 Bytes für ConUnit
7070          moveq   #1,d1               ; d1 := 1 = 'public'
7072          move.l  a6,-(a7)            ; a6 retten
7074          movea.l 26(a6),a6           ; a6 := SysBase
7078          jsr     -C6(a6)             ; ---> AllocMem
707C          movea.l (a7)+,a6            ; a6 wiederherstellen
707E          tst.l   d0                  ; Speicherbereich reserviert?
7080          beq     FE711A              ; Nein: Fehler ->
7084          movea.l d0,a2               ; a2 := d0 -> ConUnit
```

```
7Ø86          lea     BØ(pc),a1          ; a1 := FE7138 -> Tabelle f InitStruct
7Ø8A          move.l  #2A5,dØ            ; dØ := 677 = Länge von ConUnit
7Ø9Ø          move.l  a6,-(a7)           ; a6 retten
7Ø92          movea.l 26(a6),a6          ; a6 := SysBase
7Ø96          jsr     -4E(a6)            ; ---> InitStruct
7Ø9A          movea.l (a7)+,a6           ; a6 wiederherstellen
7Ø9C          lea     14(a2),aØ          ; aØ -> mp_MsgList im MsgPort
7ØAØ          move.l  aØ,(aØ)            ; Macro NewList
7ØA2          addq.l  #4,(aØ)            ; :
7ØA4          clr.l   4(aØ)              ; :
7ØA8          move.l  aØ,8(aØ)           ; :
7ØAC          move.l  28(a3),22(a2)      ; cu_Window := io_Data
7ØB2          lea     98(a6),aØ          ; aØ -> cd_DefaultKeyMap
7ØB6          lea     42(a2),a1          ; a1 -> cu_KeyMapStruct
7ØBA          moveq   #7,dØ              ; Zähler für Kopierschleife
7ØBC FE7ØBC   move.l  (aØ)+,(a1)+        ; DefaultKeyMap in ConUnit kopieren
7ØBE          dbra    dØ,FE7ØBC          ; insgesamt 8 Adressen
7ØC2          move.l  a2,18(a3)          ; ConUnit als io_Unit in IOR eintragen
7ØC6          lea     3C(a6),aØ          ; aØ -> ConUnits List Header
7ØCA          movea.l a2,a1              ; a1 := a2 -> ConUnit
7ØCC          move.l  (aØ),dØ            ; Macro AddHead
7ØCE          move.l  a1,(aØ)            ; :
7ØDØ          movem.l dØ/aØ,(a1)         ; :
7ØD4          movea.l dØ,aØ              ; :
7ØD6          move.l  a1,4(aØ)           ; :
7ØDA          bsr     FE6994             ; ---> Kursormuster initialisieren
7ØDE          bsr     FE7524             ; ---> ConUnit und Window initialisieren
7ØE2          bsr     FE6A4E             ; ---> Kursor setzen
7ØE6          tst.l   38(a6)             ; Intuition bereits geöffnet?
7ØEA          bne.s   FE71Ø2             ; Ja: ->
7ØEC          lea     38(pc),a1          ; a1 := FE7126 -> 'intuition.library'
7ØFØ          moveq   #Ø,dØ              ; Version beliebig
7ØF2          move.l  a6,-(a7)           ; a6 retten
7ØF4          movea.l 26(a6),a6          ; a6 := SysBase
7ØF8          jsr     -228(a6)           ; ---> OpenLibrary
7ØFC          movea.l (a7)+,a6           ; a6 wiederherstellen
7ØFE          move.l  dØ,38(a6)          ; IntuiBase speichern
71Ø2 FE71Ø2   move.l  4E(a6),dØ          ; dØ -> Aktives Window
71Ø6          beq.s   FE71ØE             ; Window nicht aktiv: ->
71Ø8          cmp.l   22(a2),dØ          ; = cu_Window?
```

```
710C          bne.s   FE7112              ; nein: ->
710E FE710E   move.l  a2,4A(a6)           ; a2 als Aktive ConUnit speichern
7112 FE7112   moveq   #0,d0               ; d0 := 0
7114 FE7114   movem.l (a7)+,d2-d3/a2-a4   ; Register wiederherstellen
7118          rts
711A
711A FE711A   moveq   #-1,d0              ; Fehlerkode
711C          clr.l   14(a3)              ; io_Device im IOR löschen
7120          move.b  d0,1F(a3)           ; io_Error := Fehlerkode
7124          bra.s   FE7114              ; ---> Abschluß
7126
7126 FE7126   DC.B    'intuition.library',0
7138
7138 ;--------------------------------   Tabelle für InitStruct (ConUnit)
7138
7138 FE7138   DC.W    C000                ; InitLong
713A          DC.W    A                   ; lib_Name
713C          DC.L    FE5128              ; -> 'console.device'
7140          DC.W    E000                ; InitByte
7142          DC.W    8                   ; ln_Type
7144          DC.B    4                   ; = nt_MsgPort
7145          DC.B    0
7146          DC.W    C000                ; InitLong            \
7148          DC.W    A                   ; lib_Name             | warum zweimal??
714A          DC.L    FE5128              ; -> 'console.device  /
714E          DC.W    E000                ; InitByte
7150          DC.W    20                  ; lh_Type
7152          DC.B    5                   ; = nt_Message
7153          DC.B    0
7154          DC.W    E000                ; InitByte
7156          DC.W    2A3                 ; Kursorsperre
7158          DC.B    FF                  ; Kursor entsperrt
7159          DC.B    0
715A          DC.W    D000                ; InitWord
715C          DC.W    12C                 ; ConUnit-Puffer Länge
715E          DC.W    100                 ; = 256
7160          DC.W    D000                ; InitWord
7162          DC.W    25E                 ; ESC-Puffer Länge
7164          DC.W    8                   ; = 8
7166          DC.W    D000                ; InitWord
```

```
7168          DC.W    26C                ; CSI-Puffer Länge
716A          DC.W    32                 ; = 50
716C          DC.W    0
716E
716E ;--------------------------------   Close
716E
716E FE716E   movem.l a2-a3,-(a7)        ; Register retten
7172          movea.l a1,a3              ; a3 := a1 -> IORequest
7174          movea.l 18(a3),a2          ; a2 := io_Unit
7178          cmpa.w  #-1,a2             ; io_Unit = -1?
717C          beq.s   FE71B4             ; ja: fertig ->
717E          movea.l a2,a1              ; a1 := a2 -> ConUnit
7180          lea     3C(a6),a0          ; a0 -> ConUnits List Header
7184          move.l  a6,-(a7)           ; a6 retten
7186          movea.l 26(a6),a6          ; a6 := SysBase
718A          jsr     -FC(a6)            ; ---> Remove
718E          movea.l (a7)+,a6           ; a6 wiederherstellen
7190          cmpa.l  4A(a6),a2          ; a2 = Aktive ConUnit?
7194          bne.s   FE719A             ; nein: ->
7196          clr.l   4A(a6)             ; Aktive ConUnit löschen
719A FE719A   bsr     FE699E             ; ---> rts
719E          movea.l a2,a1              ; a1 := a2 -> ConUnit
71A0          move.l  #2A5,d0            ; d0 := 677
71A6          move.l  a6,-(a7)           ; a6 retten
71A8          movea.l 26(a6),a6          ; a6 := SysBase
71AC          jsr     -D2(a6)            ; ---> FreeMem
71B0          movea.l (a7)+,a6           ; a6 wiederherstellen
71B2          movea.l a3,a1              ; a1 := a3 -> IORequest
71B4 FE71B4   movem.l (a7)+,a2-a3        ; Register wiederherstellen
71B8          bra     FE52F6             ; ---> Close
71BC
71BC ;--------------------------------   RawKeyConvert
71BC
71BC FE71BC   movem.l d2-d3/a2,-(a7)     ; Register retten
71C0          move.l  d1,d2              ; d2 := d1 = Länge des Puffers
71C2          moveq   #0,d3              ; d3 := 0
71C4          move.l  a2,d0              ; d0 := a2 -> KeyMap
71C6          bne.s   FE71CC             ; KeyMap zugeordnet: ->
71C8          lea     98(a6),a2          ; a2 -> DefaultKeyMap
71CC FE71CC   cmpi.b  #1,4(a0)           ; ieclass_RawKey?
```

```
71D2          bne      FE71E8              ; Nein: ->
71D6          move.w   6(aØ),dØ            ; dØ := ie_Code
71DA          move.w   8(aØ),d1            ; d1 := ie_Qualifier
71DE          bsr.s    FE71F6              ; ---> RawKeyConversion Routine
71EØ          tst.l    dØ                  ; Pufferüberlauf?
71E2          bmi.s    FE71FØ              ; Ja: ->
71E4          sub.l    dØ,d2               ; Pufferlänge um Kodezahl vermindern
71E6          add.l    dØ,d3               ; Kodezähler um Kodezahl erhöhen
71E8 FE71E8   move.l   (aØ),dØ             ; dØ := ie_NextEvent
71EA          movea.l  dØ,aØ              ; aØ := dØ = ie_NextEvent
71EC          bne.s    FE71CC              ; Nicht leer: ->
71EE          move.w   d3,dØ               ; dØ := d3 = Zahl der Zeichen im Puffer
71FØ FE71FØ   movem.l  (a7)+,d2-d3/a2      ; Register wiederherstellen
71F4          rts
71F6
71F6 ;------ RawKeyConversion-Routine
71F6
71F6 FE71F6   movem.l  d2-d7/a2-a5,-(a7)   ; Register retten
71FA          move.w   dØ,d6               ; d6 := dØ = ie_Code
71FC          move.w   d1,d7               ; d7 := d1 = ie_Qualifier
71FE          movea.l  d2,a4               ; a4 := d2 = Pufferlänge
72ØØ          movea.l  a2,a5               ; a5 := a2 -> KeyMap
72Ø2          andi.w   #FF7F,dØ            ; Bit 7 von ie_Code löschen
72Ø6          cmpi.w   #77,dØ              ; ie_Code > $77?
72ØA          bhi      FE732C              ; Ja: Kein legaler Kode ->
72ØE          move.w   dØ,d4               ; d4 := dØ = ie_Code (Bit 7 = Ø)
721Ø          move.w   dØ,d5               ; d5 := dØ = ie_Code (Bit 7 = Ø)
7212          andi.w   #7,d4               ; d4 := ie_Code modulo 8
7216          lsr.w    #3,d5               ; d5 := ie_Code \ 8
7218          andi.w   #7,d5               ; d5 := d5 modulo 8
721C          cmpi.w   #4Ø,dØ              ; ie_Code = LowKey?
722Ø          blt.s    FE7226              ; Ja: ->
7222          lea      1Ø(a5),a5           ; a5 -> HiKeyMapTypes
7226 FE7226   btst.l   #9,d7               ; ieQualifier_Repeat gesetzt?
722A          beq.s    FE7238              ; Nein: ->
722C          movea.l  C(a5),a3            ; a3 := km_Low/HiRepeatable
723Ø          btst.b   d4,Ø(a3,d5.w)       ; Wiederholung zulässig?
7234          beq      FE732C              ; Nein: ->
7238 FE7238   movea.l  Ø(a5),a3            ; a3 := km_Low/HiKeyMapTypes
723C          move.w   dØ,d1               ; d1 := dØ = ie_Code (Bit 7 = Ø)
```

```
723E          andi.w  #3F,d1              ; Bit 6 löschen
7242          move.b  0(a3,d1.w),d2       ; d2 := Typ des ie_Codes
7246          btst.l  #7,d2               ; Nop-Kode?
724A          bne     FE732C              ; Ja: ->
724E          moveq   #0,d3               ; d3 := 0
7250          cmp.w   d6,d0               ; Taste gedrückt?
7252          beq.s   FE725E              ; Nein: ->
7254          btst.l  #3,d2               ; kcb_DownUp gesetzt?
7258          beq     FE732C              ; Nein: ->
725C          addq.w  #8,d3               ; Bit 3 in d3 setzen
725E FE725E   move.w  d7,d0               ; d0 := d7 = ie_Qualifier
7260          andi.w  #3,d0               ; ieq_LShift oder -RShift gesetzt?
7264          beq.s   FE7268              ; Nein: ->
7266          addq.w  #1,d3               ; Bit 0 in d3 setzen
7268 FE7268   move.w  d7,d0               ; d0 := d7 = ie_Qualifier
726A          andi.w  #30,d0              ; ieq_LAlt oder -RAlt gesetzt?
726E          beq.s   FE7272              ; Nein: ->
7270          addq.w  #2,d3               ; Bit 1 in d3 setzen
7272 FE7272   btst.l  #2,d7               ; ieq_CapsLock gesetzt?
7276          beq.s   FE7286              ; Nein: ->
7278          movea.l 8(a5),a3            ; a3 := km_Low/HiCapsable
727C          btst.b  d4,0(a3,d5.w)       ; Caps zulässig?
7280          beq.s   FE7286              ; Nein: ->
7282          ori.w   #1,d3               ; Bit 0 in d3 setzen
7286 FE7286   btst.l  #3,d7               ; ieq_Control gesetzt?
728A          beq.s   FE728E              ; Nein: ->
728C          addq.w  #4,d3               ; Bit 2 in d3 setzen
728E FE728E   movea.l 4(a5),a3            ; a3 := km_Low/HiKeyMap
7292          lsl.w   #2,d1               ; d1 := d1 * 4 ergibt Zeiger in KeyMap
7294          move.l  0(a3,d1.w),d5       ; d5 := 4 Keykodes aus KeyMap
7298          moveq   #0,d4               ; d4 := 0: Bytezeiger initialisieren
729A          moveq   #0,d0               ; d0 := 0: Zähler initialisieren
729C          moveq   #1,d1               ; d1 := 1: Bit-Addend initialisieren
729E FE729E   btst.l  d0,d2               ; kcb_Bit (d0) wirksam?
72A0          beq.s   FE72AA              ; Nein: ->
72A2          btst.l  d0,d3               ; ieq_Bit (d0) gesetzt?
72A4          beq.s   FE72A8              ; Nein: ->
72A6          add.w   d1,d4               ; Bit im Bytezeiger setzen
72A8 FE72A8   add.w   d1,d1               ; Bit-Addend erhöhen
72AA FE72AA   addq.w  #1,d0               ; Zähler erhöhen
```

```
72AC          cmpi.w  #4,d0               ; Schon 4 Fälle geprüft?
72B0          blt.s   FE729E              ; Nein: ->
72B2          bclr.l  #5,d2               ; kcb_Dead gesetzt?
72B6          bne     FE7348              ; Ja: ->
72BA          btst.l  #6,d2               ; kcb_String gesetzt?
72BE          bne.s   FE7316              ; Ja: ->
72C0          cmpi.b  #7,d2               ; Vanilla-Key?
72C4          beq.s   FE72DA              ; Ja: ->
72C6          cmpi.w  #4,d1               ; Höchstens 2 ieq-Bits gesetzt?
72CA          ble.s   FE72D4              ; Ja: ->
72CC FE72CC   movea.l d5,a3               ; a3 := d5 -> Tabelle
72CE          move.b  0(a3,d4.w),d5       ; d5 := Kode aus Tabelle
72D2          bra.s   FE7308              ; ---> Kode in Puffer, Abschluß
72D4
72D4 FE72D4   lsl.w   #3,d4               ; Bytezeiger mal 8 ergibt Bitzeiger
72D6          ror.l   d4,d5               ; Keykode in d5.b rotieren
72D8          bra.s   FE7308              ; ---> Kode in Puffer, Abschluß
72DA
72DA ;------ Vanilla Key
72DA
72DA FE72DA   lsl.w   #3,d4               ; Bytezeiger mal 8 ergibt Bitzeiger
72DC          ror.l   d4,d5               ; Keykode in d5.b rotieren
72DE          btst.l  #2,d3               ; ieq_Control gesetzt?
72E2          beq.s   FE7308              ; Nein: ->
72E4          moveq   #3,d0               ; d0 := 3
72E6 FE72E6   move.b  d5,d1               ; d1 := d5 = Keykode
72E8          andi.w  #C0,d1              ; Bits 6 und 7 isolieren
72EC          cmpi.w  #40,d1              ; Keykode >= $40?
72F0          beq.s   FE72FA              ; Ja: ->
72F2          ror.l   #8,d5               ; Vorigen Keykode einrotieren
72F4          dbra    d0,FE72E6           ; maximal 3 mal wiederholen ->
72F8          bra.s   FE7308              ; ---> Kode in Puffer, Abschluß
72FA
72FA FE72FA   andi.w  #1F,d5              ; Bits 5,6,7 des Keykode löschen
72FE          btst.l  #2,d3               ; ieq_Control gesetzt?
7302          bne.s   FE7308              ; Ja: ->
7304          ori.w   #80,d5              ; Bit 7 setzen
7308 FE7308   move.l  a4,d0               ; d0 := a4 = Pufferlänge
730A          beq.s   FE7312              ; = 0: Fehler ->
730C          move.b  d5,(a1)+            ; Keykode in Puffer schreiben
```

```
730E          moveq   #1,d0               ; d0 := 1: 1 Zeichen im Puffer
7310          bra.s   FE732E              ; ---> Abschluß
7312
7312 FE7312   moveq   #-1,d0              ; d0 := -1: Pufferüberlauf
7314          bra.s   FE732E              ; ---> Abschluß
7316
7316 ;------ String Key
7316
7316 FE7316   bsr.s   FE7336              ; ---> Länge u. Anfangsoffset holen
7318          cmp.l   a4,d0               ; Stringlänge > Pufferlänge?
731A          bgt.s   FE7312              ; Ja: Fehler ->
731C          ext.w   d1                  ; d1.w := String-Anfangsoffset
731E          adda.w  d1,a3               ; zur Deskriptoradresse addieren
7320          move.w  d0,d1               ; d1 := d0 = Stringlänge
7322          bra.s   FE7326              ; ---> String kopieren
7324
7324 FE7324   move.b  (a3)+,(a1)+         ; Kode in Puffer kopieren
7326 FE7326   dbra    d1,FE7324           ; wiederholen bis String kopiert ->
732A          bra.s   FE732E              ; ---> Abschluß
732C
732C FE732C   moveq   #0,d0               ; d0 := 0: Kein Zeichen im Puffer
732E FE732E   moveq   #0,d1               ; d1 := 0
7330 FE7330   movem.l (a7)+,d2-d7/a2-a5   ; Register wiederherstellen
7334          rts
7336
7336 ;------ Stringdeskriptor holen
7336
7336 FE7336   add.w   d4,d4               ; Bytezeiger verdoppeln
7338          movea.l d5,a3               ; a3 := d5 -> String Deskriptor
733A          moveq   #0,d0               ; d0 := 0
733C          moveq   #0,d1               ; d1 := 0
733E          move.b  0(a3,d4.w),d0       ; d0 := Länge des Strings
7342          move.b  1(a3,d4.w),d1       ; d1 := Anfangsoffset des Strings
7346          rts
7348
7348 ;------ Dead Key
7348
7348 FE7348   bsr.s   FE7336              ; ---> Deskriptor holen
734A          btst.l  #3,d0               ; Bit 3 im Längenbyte gesetzt?
734E          bne.s   FE736A              ; Ja: ->
```

```
7350          btst.l  #0,d0               ; Bit 0 im Längenbyte gesetzt?
7354          bne.s   FE736E              ; Ja: ->
7356          btst.l  #6,d3               ; ieq_LCommand (LAmiga) gesetzt?
735A          bne.s   FE7360              ; Ja: ->
735C          move.b  d1,d5               ; d5 := d1 = Kode aus Keymap
735E          bra.s   FE7308              ; ---> Kode in Puffer
7360
7360 FE7360   move.l  a3,d5               ; d5 := a3 -> Deskriptor
7362          ext.w   d1                  ; d1.w = Anfangsoffset
7364          add.w   d1,d5               ; d5 := Adresse des Strings
7366          moveq   #0,d4               ; d4 := 0
7368          bra.s   FE7316              ; ---> String in Puffer
736A
736A FE736A   moveq   #0,d0               ; d0 := 0: Kein Zeichen im Puffer
736C          bra.s   FE7330              ; ---> Ausgang
736E
736E FE736E   move.l  a4,d0               ; d0 := a4 (Pufferlänge)
7370          beq.s   FE7312              ; = 0: Rekursion stop ->
7372          move.l  a3,d5               ; d5 := a3 -> Deskriptor
7374          ext.w   d1                  ; d1.w = Anfangsoffset
7376          add.w   d1,d5               ; d5 -> Tabelleneintrag
7378          moveq   #0,d4               ; d4 := 0
737A          moveq   #0,d0               ; d0 := 0
737C          move.b  A(a0),d0            ; d0 := voriger Tastenkode
7380          moveq   #0,d1               ; d1 := 0
7382          move.b  B(a0),d1            ; d1 := voriger Qualifier
7386          moveq   #0,d2               ; d2 := 0 (verhindert Rekursion)
7388          bsr     FE71F6              ; ---> RawKeyConversion Routine
738C          tst.w   d1                  ; Zeichen im Puffer?
738E          beq.s   FE73B4              ; ja: Fertig ->
7390          move.w  d1,d4               ; d4 := d1 = Zweites Deskriptorbyte
7392          andi.w  #F,d4               ; Bits 0 bis 3 isolieren
7396          lsr.w   #4,d1               ; Bits 4 bis 7 nach unten schieben
7398          beq.s   FE73B4              ; = 0: Keine Kombination möglich ->
739A          mulu    d1,d4               ; d4 := d1 * d4
739C          moveq   #0,d0               ; d0 := 0
739E          move.b  C(a0),d0            ; d0 := vorvoriger Tastenkode
73A2          moveq   #0,d1               ; d1 := 0
73A4          move.b  D(a0),d1            ; d1 := vorvoriger Qualifier
73A8          moveq   #0,d2               ; d2 := 0 (verhindert Rekursion)
```

```
73AA          bsr     FE71F6              ; ---> RawKeyConversion Routine
73AE          andi.w  #F,d1               ; Bits 0 bis 3 isolieren
73B2          add.w   d1,d4               ; Ergebnis zu d4 addieren
73B4 FE73B4   btst.l  #6,d3               ; ieq_LCommand (Amiga links)?
73B8          bne     FE7316              ; ja: String in Puffer ->
73BC          bra     FE72CC              ; ---> Kode aus Tabelle in Puffer
73C0
73C0 ;--------------------------------   cd_Read
73C0
73C0 FE73C0   movem.l d2-d3/a2-a5,-(a7)   ; Register retten
73C4          movea.l a1,a3               ; a3 := a1 -> IORequest
73C6          move.l  a6,-(a7)            ; a6 retten
73C8          movea.l 26(a6),a6           ; a6 := SysBase
73CC          jsr     -84(a6)             ; ---> Forbid
73D0          movea.l (a7)+,a6            ; a6 wiederherstellen
73D2          bset.b  #6,1E(a3)           ; Wiederholungssperre setzen
73D8          bne     FE7418              ; war schon gesetzt: ->
73DC          moveq   #0,d2               ; d2 := 0
73DE          move.l  24(a3),d3           ; d3 := io_Length
73E2          beq     FE742A              ; = 0: Fehler ->
73E6          movea.l 18(a3),a2           ; a2 := io_Unit
73EA          lea     128(a2),a4          ; a4 -> ConUnit-Puffer
73EE          movea.l a4,a0               ; a0 := a4 -> ConUnit-Puffer
73F0          bsr     FE6894              ; ---> d0 := Länge des Pufferinhalts
73F4          tst.w   d0                  ; Länge = 0?
73F6          beq     FE7412              ; ja: Nichts zu lesen ->
73FA          movea.l 28(a3),a5           ; a5 := io_Data
73FE FE73FE   movea.l a4,a0               ; a0 := a4 -> ConUnit-Puffer
7400          bsr     FE6854              ; ---> Byte aus ConUnit-Puffer lesen
7404          tst.w   d0                  ; Byte gelesen?
7406          bmi.s   FE7434              ; nein: ->
7408          move.b  d0,(a5)+            ; Byte in io_Data-Bereich kopieren
740A          addq.l  #1,d2               ; Bytezähler erhöhen
740C          subq.l  #1,d3               ; io_Length erniedrigen
740E          ble.s   FE7434              ; Unterlauf: ->
7410          bra.s   FE73FE              ; ---> Loop
7412
7412 FE7412   andi.b  #BE,1E(a3)          ; Wiederholungssperre löschen
7418 FE7418   move.l  a6,-(a7)            ; a6 retten
741A          movea.l 26(a6),a6           ; a6 := SysBase
```

```
741E          jsr     -8A(a6)              ; ---> Permit
7422          movea.l (a7)+,a6             ; a6 wiederherstellen
7424          movem.l (a7)+,d2-d3/a2-a5    ; Register wiederherstellen
7428          rts
742A
742A FE742A   moveq   #-4,d0               ; d0 := ioerr_BadLength
742C          movea.l a3,a1                ; a1 := a3 -> IORequest
742E          bsr     FE5226               ; ---> IORequest beantworten
7432          bra.s   FE7440               ; ---> Weiterlesen oder Abschluß
7434
7434 FE7434   move.l  d2,20(a3)            ; io_Actual := Zahl der Kodes
7438          movea.l a3,a1                ; a1 := a3 -> IORequest
743A          moveq   #0,d0                ; d0 := ok-Kode
743C          bsr     FE5226               ; ---> IORequest beantworten
7440 FE7440   movea.l 14(a2),a1            ; a1 := io_Device
7444          tst.l   (a1)                 ; (a1) = 0?
7446          beq.s   FE7418               ; ja: ->
7448          bsr     FE73C0               ; ---> cd_Read
744C          bra.s   FE7418               ; ---> Abschluß
744E
744E          DC.W    0
7450
7450 ;--------------------------------   Tabulator-Routinen
7450
7450 ;------ TabStops im Abstand 8 setzen
7450
7450 FE7450   moveq   #0,d0                ; d0 := 0
7452          moveq   #4E,d1               ; d1 := 78 = Maximale TabStop-Zahl
7454          lea     62(a2),a0            ; a0 -> cu_TabStops
7458 FE7458   move.w  d0,(a0)+             ; TabOffset in Tabelle schreiben
745A          addq.w  #8,d0                ; TabOffset um 8 erhöhen
745C          dbra    d1,FE7458            ; wiederholen bis d1 = 0 ->
7460          move.w  #-1,(a0)             ; Endemarke anfügen
7464          rts
7466
7466 ;------ Tabulator vorwärts
7466
7466 FE7466   bsr.s   FE749E               ; ---> Nächste Tabposition suchen
7468          beq.s   FE746C               ; TabStop an aktueller Position: ->
746A          subq.l  #2,a0                ; auf letzten TabStop zurücksetzen
```

```
746C FE746C  move.w  (aØ),dØ          ; dØ := nächste Tabposition
746E         cmp.w   2A(a2),dØ        ; > Maximale Position?
7472         bhi.s   FE747C           ; ja: ->
7474         move.w  dØ,26(a2)        ; cu_XCP := nächste Tabposition
7478 FE7478  bra     FE6A66           ; ---> Kursor umsetzen
747C
747C FE747C  btst.b  #6,124(a2)       ; AutoWrap-Flag gesetzt?
7482         bne.s   FE748C           ; ja: ->
7484         move.w  2A(a2),26(a2)    ; cu_XCP := cu_XMax
748A         bra.s   FE7478           ; ---> Kursor umsetzen
748C
748C FE748C  clr.w   26(a2)           ; cu_XCP := Ø
749Ø         moveq   #1,dØ            ; dØ := 1
7492         bsr     FE6AA6           ; ---> Kursor dØ Zeilen abwärts
7496         bset.b  #Ø,2A4(a2)       ; Zeilenverkettungsflag setzen
749C         rts
749E
749E ;------ Nächsten TabStop suchen
749E
749E FE749E  move.w  26(a2),dØ        ; dØ := cu_XCP = aktuelle Position
74A2         moveq   #FF,d1           ; Zähler hochsetzen
74A4         lea     62(a2),aØ        ; aØ -> cu_TabStops
74A8 FE74A8  cmp.w   (aØ)+,dØ         ; dØ mit TabStops vergleichen
74AA         dbls    d1,FE74A8        ; bis dØ <= TabStop ->
74AE FE74AE  rts
74BØ
74BØ ;------ Tabulator rückwärts
74BØ
74BØ FE74BØ  tst.w   26(a2)           ; cu_XCP = Ø?
74B4         beq.s   FE74C4           ; ja: ->
74B6         bsr.s   FE749E           ; ---> Nächsten TabStop suchen
74B8         subq.l  #4,aØ            ; um 2 TabStops zurücksetzen
74BA         move.w  (aØ),dØ          ; dØ := vorhergehender TabStop
74BC         move.w  dØ,26(a2)        ; als aktuelle Position speichern
74CØ         bra     FE6A66           ; ---> Kursor umsetzen
74C4
74C4 FE74C4  btst.b  #6,124(a2)       ; AutoWrap-Flag gesetzt?
74CA         beq.s   FE74AE           ; nein: fertig ->
74CC         moveq   #1,dØ            ; dØ := 1
74CE         bsr     FE6A82           ; ---> Kursor dØ Zeilen aufwärts
```

```
74D2          move.w  2A(a2),d0           ; d0 := cu_XMax
74D6          addq.w  #1,d0               ; + 1
74D8          move.w  d0,26(a2)           ; als aktuelle Position speichern
74DC          bra.s   FE74B0              ; ---> Tab rückwärts
74DE
74DE ;------ TabStop an aktueller Position setzen
74DE
74DE FE74DE   bsr.s   FE749E              ; ---> Nächsten TabStop suchen
74E0          beq.s   FE74AE              ; Aktuelle Position = TabStop: ->
74E2          lea     100(a2),a0          ; a0 -> Ende der TabStop-Tabelle
74E6          move.w  #-1,(a0)            ; Endemarke einschreiben
74EA          subq.l  #2,a0               ; auf vorherigen TabStop zurücksetzen
74EC          addi.w  #4F,d1              ; MaxTabs zum Suchzähler addieren
74F0          bra.s   FE74F6              ; --->
74F2
74F2 FE74F2   move.w  -(a0),2(a0)         ; TabStops umkopieren
74F6 FE74F6   dbra    d1,FE74F2           ; Loop ->
74FA          move.w  26(a2),(a0)         ; cu_XCP an Tabellenanfang schreiben
74FE          rts
7500
7500 ;------ TabStop an aktueller Position löschen
7500
7500 FE7500   tst.w   26(a2)              ; cu_XCP = 0?
7504          beq.s   FE74AE              ; ja: fertig ->
7506          bsr.s   FE749E              ; ---> Nächsten TabStop suchen
7508          bne.s   FE74AE              ; Aktuelle Position ist nicht TabStop ->
750A          addi.w  #4F,d1              ; MaxTabs zum Suchzähler addieren
750E FE750E   move.w  (a0),-2(a0)         ; TabStops umkopieren
7512          addq.l  #2,a0               ; Tabellenzeiger erhöhen
7514          dbra    d1,FE750E           ; Loop ->
7518          rts
751A
751A ;------ Alle TabStops löschen
751A
751A FE751A   move.l  #-1,62(a2)          ; Endemarke an Tabellenanfang
7522          rts
7524
7524 ;--------------------------------   ConUnit und Window initialisieren
7524
7524 FE7524   bsr     FE6A32              ; ---> Kursor löschen
```

```
7528            bsr     FE745Ø              ; ---> TabStops im Abstand 8 setzen
752C            move.l  #FE7A6C,256(a2)     ; Vektor für cd_Write -> Textausgabe
7534            move.w  25A(a2),25C(a2)     ; ESC-Puffer löschen
753A            move.w  268(a2),26A(a2)     ; CSI-Puffer löschen
754Ø            clr.w   2AØ(a2)             ; cd_Write Branchparameter 'normal'
7544            clr.l   26(a2)              ; cu_XCP, cu_YCP := Ø
7548            clr.l   3A(a2)              ; cu_XMinShrink, cu_YMinShrink := Ø
754C            move.w  #27ØF,36(a2)        ; cu_XRExtant := 9999
7552            move.w  #27ØF,38(a2)        ; cu_YRExtant := 9999
7558            move.b  #2,2A4(a2)          ; Kursor einschalten, Flags löschen
755E            moveq   #2,dØ               ; dØ := 2 = Zähler für Löschschleife
756Ø            lea     122(a2),aØ          ; aØ -> Mode Flags
7564 FE7564     clr.b   (aØ)+               ; ModeFlag-Byte löschen
7566            dbra    dØ,FE7564           ; 2 mal wiederholen ->
756A            bset.b  #4,124(a2)          ; LineFeed-Flag setzen
757Ø            bset.b  #5,124(a2)          ; AutoScroll-Flag setzen
7576            bset.b  #6,124(a2)          ; AutoWrap-Flag setzen
757C            bsr     FE66D8              ; ---> Lock Display
758Ø            movea.l 22E(a2),a1          ; a1 := wd_RPort
7584            move.w  25Ø(a2),dØ          ; dØ := rp_TxWidth
7588            move.w  dØ,3C(a1)           ; zurück in RastPort
758C            move.w  254(a2),d1          ; d1 := rp_TxSpacing
759Ø            move.w  d1,4Ø(a1)           ; zurück in RastPort
7594            add.w   dØ,d1               ; + rp_TxWidth
7596            move.w  d1,2E(a2)           ; als cu_XRSize speichern
759A            move.w  24E(a2),d1          ; d1 := rp_TxHeight
759E            move.w  d1,3A(a1)           ; zurück in RastPort
75A2            move.w  d1,3Ø(a2)           ; als cu_YRSize speichern
75A6            move.l  248(a2),34(a1)      ; rp_Font
75AC            move.w  24C(a2),38(a1)      ; rp_AlgoStyle, rp_TxFlags
75B2            move.w  252(a2),3E(a1)      ; rp_TxBaseline
75B8            bsr     FE8Ø78              ; ---> Plain Style
75BC            bsr     FE76BØ              ; ---> Ausgabebereich initialisieren
75CØ            move.w  32(a2),24(a1)       ; rp_cp_x := cu_XROrigin
75C6            move.w  34(a2),d1           ; d1 := cu_YROrigin
75CA            add.w   3E(a1),d1           ; + rp_TxBaseline
75CE            move.w  d1,26(a1)           ; rp_cp_y := d1
75D2            bsr     FE6A66              ; ---> Kursor umsetzen
75D6            bsr     FE6A4E              ; ---> Kursor setzen
75DA            bsr     FE678Ø              ; ---> Unlock Display
```

```
75DE          rts
75E0
75E0 ;----------------------------   Breite/Höhe des Ausgabebereichs ermitteln
75E0
75E0 FE75E0   movea.l 22(a2),a0           ; a0 := cu_Window
75E4          move.l  18(a0),d0           ; d0 := wd_Flags
75E8          andi.l  #400,d0             ; GZZ-Window?
75EE          beq.s   FE763E              ; nein: ->
75F0          btst.b  #6,2A4(a2)          ; Linksoffset gesetzt?
75F6          bne.s   FE75FC              ; ja: ->
75F8          clr.w   32(a2)              ; cu_XROrigin := 0
75FC FE75FC   move.w  8(a0),d0            ; d0 := wd_Width
7600          move.b  36(a0),d1           ; d1 := wd_BorderLeft
7604          ext.w   d1                  ; d1 auf Wort erweitern
7606          sub.w   d1,d0               ; d0 := wd_Width - wd_BorderLeft
7608          move.b  38(a0),d1           ; d1 := wd_BorderRight
760C          ext.w   d1                  ; d1 auf Wort erweitern
760E          sub.w   d1,d0               ; d0 := d0 - wd_BorderRight
7610          sub.w   32(a2),d0           ; d0 := d0 - cu_XROrigin
7614          move.w  d0,d6               ; d6 := d0 = Breite des Ausgabebereichs
7616          btst.b  #7,2A4(a2)          ; Topoffset gesetzt?
761C          bne.s   FE7622              ; ja: ->
761E          clr.w   34(a2)              ; cu_YROrigin := 0
7622 FE7622   move.w  A(a0),d0            ; d0 := wd_Height
7626          move.b  37(a0),d1           ; d1 := wd_BorderTop
762A          ext.w   d1                  ; d1 auf Wort erweitern
762C          sub.w   d1,d0               ; d0 := wd_Height - wd_BorderTop
762E          move.b  39(a0),d1           ; d1 := wd_BorderBottom
7632          ext.w   d1                  ; d1 auf Wort erweitern
7634          sub.w   d1,d0               ; d0 := d0 - wd_BorderBottom
7636          sub.w   32(a2),d0           ; d0 := d0 - cu_XROrigin (Fehler! Y!!)
763A          move.w  d0,d7               ; d7 := d0 = Höhe des Ausgabebereichs
763C          bra.s   FE7686              ; --->
763E
763E FE763E   btst.b  #6,2A4(a2)          ; Linksoffset gesetzt?
7644          bne.s   FE7650              ; ja: ->
7646          move.b  36(a0),d1           ; d1 := wd_BorderLeft
764A          ext.w   d1                  ; d1 auf Wort erweitern
764C          move.w  d1,32(a2)           ; cu_XROrigin := wd_BorderLeft
7650 FE7650   move.w  8(a0),d0            ; d0 := wd_Width
```

```
7654          sub.w   32(a2),d0           ; d0 := wd_Width - cu_XROrigin
7658          move.b  38(a0),d1           ; d1 := wd_BorderRight
765C          ext.w   d1                  ; d1 auf Wort erweitern
765E          sub.w   d1,d0               ; d0 := d0 - wd_BorderRight
7660          move.w  d0,d6               ; d6 := d0 = Breite des Ausgabebereichs
7662          btst.b  #7,2A4(a2)          ; Topoffset gesetzt?
7668          bne.s   FE7674              ; ja: ->
766A          move.b  37(a0),d1           ; d1 := wd_BorderTop
766E          ext.w   d1                  ; d1 auf Wort erweitern
7670          move.w  d1,34(a2)           ; cu_YROrigin := wd_BorderTop
7674 FE7674   move.w  A(a0),d0            ; d0 := wd_Height
7678          sub.w   34(a2),d0           ; d0 := wd_Height-cu_YROrigin
767C          move.b  39(a0),d1           ; d1 := wd_BorderBottom
7680          ext.w   d1                  ; d1 auf Wort erweitern
7682          sub.w   d1,d0               ; d0 := d0-wd_BorderBottom
7684          move.w  d0,d7               ; d7 := d0 = Höhe des Ausgabebereichs
7686 FE7686   rts
7688
7688 ;-------------------------------   Ausgaberest erfassen
7688
7688 FE7688   movem.l d6-d7,-(a7)         ; Register retten
768C          bsr     FE75E0              ; ---> Ausgabebéreichsgröße bestimmen
7690          cmp.w   3A(a2),d6           ; Breite < cu_XMinShrink?
7694          bge.s   FE769A              ; nein: ->
7696          move.w  d6,3A(a2)           ; cu_XMinShrink := Breite
769A FE769A   cmp.w   3C(a2),d7           ; Höhe < cu_YMinShrink?
769E          bge.s   FE76A4              ; nein: ->
76A0          move.w  d7,3C(a2)           ; cu_YMinShrink := Höhe
76A4 FE76A4   bset.b  #2,2A4(a2)          ; ShrinkFlag setzen
76AA          movem.l (a7)+,d6-d7         ; Register wiederherstellen
76AE          rts
76B0
76B0 ;-------------------------------   Ausgabebereich initialisieren
76B0
76B0 FE76B0   movem.l d2-d7,-(a7)         ; Register retten
76B4          bsr     FE75E0              ; ---> Ausgabebereichsgröße bestimmen
76B8          movea.l 22(a2),a0           ; a0 -> cu_Window
76BC          move.l  18(a0),d0           ; d0 := wd_Flags
76C0          andi.l  #C0,d0              ; Refresh-Bits isolieren
76C6          cmpi.l  #0,d0               ; Smart_Refresh?
```

```
76CC          bne.s   FE7746          ; nein: ->
76CE          moveq   #0,d0           ; d0 := 0
76D0          move.w  3A(a2),d0       ; d0 := cu_XMinShrink
76D4          move.w  2E(a2),d1       ; d1 := cu_XRSize
76D8          divu    d1,d0           ; d0 := Minimale Spaltenzahl
76DA          move.w  d0,d4           ; d4 := d0 = Min. Spaltenzahl
76DC          mulu    2E(a2),d0       ; d0 := Spaltenzahl*Zeichenbreite
76E0          add.w   32(a2),d0       ; + cu_XROrigin
76E4          move.w  d6,d2           ; d2 := d6 = Breite des Ausgabebereichs
76E6          add.w   32(a2),d2       ; + cu_XROrigin
76EA          subq.w  #1,d2           ; -1
76EC          cmp.w   36(a2),d2       ; > cu_XRExtant?
76F0          ble.s   FE76F6          ; nein: ->
76F2          move.w  36(a2),d2       ; sonst durch cu_XRExtant ersetzen
76F6 FE76F6   move.w  34(a2),d1       ; d1 := cu_YROrigin
76FA          move.w  d7,d3           ; d3 := d7 = Höhe des Ausgabebereichs
76FC          add.w   34(a2),d3       ; + cu_YROrigin
7700          subq.w  #1,d3           ; -1
7702          cmp.w   38(a2),d3       ; > cu_YRExtant?
7706          ble.s   FE770C          ; nein: ->
7708          move.w  38(a2),d3       ; sonst durch cu_YRExtant ersetzen
770C FE770C   cmp.w   d0,d2           ; xmax < xmin?
770E          blt.s   FE771C          ; ja: ->
7710          cmp.w   d1,d3           ; ymax < ymin?
7712          blt.s   FE771C          ; ja: ->
7714          movea.l 22E(a2),a1      ; a1 := wd_RPort
7718          bsr     FE6930          ; ---> Rechteck im Display löschen
771C FE771C   moveq   #0,d1           ; d1 := 0
771E          move.w  3C(a2),d1       ; d1 := cu_YMinShrink
7722          move.w  30(a2),d0       ; d0 := cu_YRSize
7726          divu    d0,d1           ; d1 := Minimale Zeilenzahl
7728          move.w  d1,d5           ; d5 := d1 = Minimale Zeilenzahl
772A          mulu    30(a2),d1       ; d1 := Minimale Fensterhöhe
772E          add.w   34(a2),d1       ; + cu_YROrigin
7732          move.w  32(a2),d0       ; d0 := cu_XROrigin
7736          cmp.w   d0,d2           ; xmax < xmin?
7738          blt.s   FE7746          ; ja: ->
773A          cmp.w   d1,d3           ; ymax < ymin?
773C          blt.s   FE7746          ; ja: ->
773E          movea.l 22E(a2),a1      ; a1 := wd_RPort
```

```
7742          bsr     FE6930              ; ---> Rechteck im Display löschen
7746 FE7746   btst.b  #4,2A4(a2)          ; Zeilenlänge gesetzt?
774C          bne.s   FE775E              ; ja: ->
774E          moveq   #0,d0               ; d0 := 0
7750          move.w  d6,d0               ; d0 := d6 = Breite des Ausgabebereichs
7752          move.w  2E(a2),d1           ; d1 := cu_XRSize
7756          divu    d1,d0               ; d0 := Spaltenzahl
7758          subq.w  #1,d0               ; - 1
775A          move.w  d0,2A(a2)           ; als cu_XMax speichern
775E FE775E   move.w  2A(a2),d0           ; d0 := cu_XMax
7762          addq.w  #1,d0               ; + 1
7764          mulu    2E(a2),d0           ; * cu_XRSize
7768          subq.w  #1,d0               ; - 1
776A          add.w   32(a2),d0           ; + cu_XROrigin
776E          move.w  d0,36(a2)           ; als cu_XRExtant speichern
7772          btst.b  #5,2A4(a2)          ; Seitenlänge gesetzt?
7778          bne.s   FE778A              ; ja: ->
777A          moveq   #0,d0               ; d0 := 0
777C          move.w  d7,d0               ; d0 := d7 = Höhe des Ausgabebereichs
777E          move.w  30(a2),d1           ; d1 := cu_YRSize
7782          divu    d1,d0               ; d0 := Zeilenzahl
7784          subq.w  #1,d0               ; - 1
7786          move.w  d0,2C(a2)           ; als cu_YMax speichern
778A FE778A   move.w  2C(a2),d0           ; d0 := cu_YMax
778E          addq.w  #1,d0               ; + 1
7790          mulu    30(a2),d0           ; * cu_YRSize
7794          subq.w  #1,d0               ; - 1
7796          add.w   34(a2),d0           ; + cu_YROrigin
779A          move.w  d0,38(a2)           ; als cu_YRExtant speichern
779E          move.w  26(a2),d0           ; d0 := cu_XCP
77A2          cmp.w   d0,d4               ; Min. Spaltenzahl > akt.Position?
77A4          ble.s   FE77B4              ; nein: ->
77A6          move.w  28(a2),d1           ; d1 := cu_YCP
77AA          cmp.w   d1,d5               ; Min. Zeilenzahl > aktuelle Zeile?
77AC          ble.s   FE77B4              ; nein: ->
77AE          bsr     FE6A32              ; ---> Kursor löschen
77B2          bra.s   FE77B8              ; --->
77B4
77B4 FE77B4   addq.b  #1,2A2(a2)          ; Kursorsperre erhöhen
77B8 FE77B8   movea.l 22(a2),a0           ; a0 := cu_Window
```

```
77BC          move.l  18(aØ),dØ          ; dØ := wd_Flags
77CØ          andi.l  #8Ø,dØ             ; Super_BitMap?
77C6          bne.s   FE77FA             ; ja: ->
77C8          move.w  26(a2),dØ          ; dØ := cu_XCP
77CC          cmp.w   2A(a2),dØ          ; dØ > cu_XMax?
77DØ          ble.s   FE77DA             ; nein: ->
77D2          addq.w  #1,28(a2)          ; cu_YCP inkrementieren
77D6          clr.w   26(a2)             ; cu_XCP := Ø
77DA FE77DA   move.w  28(a2),dØ          ; dØ := cu_YCP
77DE          cmp.w   2C(a2),dØ          ; dØ > cu_YMax?
77E2          ble.s   FE77F6             ; nein: ->
77E4          clr.w   26(a2)             ; cu_XCP := Ø
77E8          move.w  2C(a2),28(a2)      ; cu_YCP := cu_YMax
77EE          moveq   #Ø,dØ              ; dØ := Ø = Dx
77FØ          moveq   #1,d1              ; d1 := 1 = Dy
77F2          bsr     FE6B3E             ; ---> Scroll Display
77F6 FE77F6   bsr     FE6A4E             ; ---> Kursor setzen
77FA FE77FA   move.w  #27ØF,3A(a2)       ; cu_XMinShrink := 9999
78ØØ          move.w  #27ØF,3C(a2)       ; cu_YMinShrink := 9999
78Ø6          movem.l (a7)+,d2-d7        ; Register wiederherstellen
78ØA          rts
78ØC
78ØC ;--------------------------------   cd_Write
78ØC
78ØC FE78ØC   movem.l d2-d6/a2-a4,-(a7)  ; Register retten
781Ø          movea.l a1,a3              ; a3 := a1 -> IORequest
7812          movea.l 18(a3),a2          ; a2 := io_Unit
7816          clr.l   2Ø(a3)             ; io_Actual := Ø
781A          bsr     FE66D8             ; ---> Lock Display
781E          move.l  24(a3),dØ          ; dØ := io_Length
7822          beq.s   FE785C             ; = Ø: ->
7824          cmpi.l  #-1,dØ             ; = -1?
782A          bne.s   FE7842             ; nein: ->
782C          moveq   #FF,dØ             ; Zähler für Suche hochsetzen
782E          movea.l 28(a3),aØ          ; aØ := io_Data
7832 FE7832   tst.b   (aØ)+              ; Endekode Ø suchen
7834          dbeq    dØ,FE7832          ; wiederholen bis Ø gefunden ->
7838          neg.l   dØ                 ; dØ := Zeichenzahl+1
783A          subq.w  #1,dØ              ; - 1
783C          move.l  dØ,24(a3)          ; Textlänge als io_Length speichern
```

```
784Ø          beq.s   FE785C              ; = Ø: ->
7842 FE7842   movea.l 28(a3),aØ           ; aØ := io_Data
7846          clr.w   dØ                  ; dØ := Ø
7848          move.b  (aØ),dØ             ; dØ := Kode aus Text
784A          move.w  2AØ(a2),d1          ; d1 := Branch-Parameter
784E          lsl.w   #2,d1               ; mal 4
785Ø          movea.l 58(pc,d1.w),a1      ; ---> FE78AA: Vektortabelle
7854          jsr     (a1)                ; ---> Funktion aufrufen
7856          tst.l   24(a3)              ; io_Length = Ø?
785A          bne.s   FE7842              ; nein: ->
785C FE785C   bsr     FE678Ø              ; ---> Unlock Display
786Ø          move.l  a6,-(a7)            ; a6 retten
7862          movea.l 26(a6),a6           ; a6 := SysBase
7866          jsr     -84(a6)             ; ---> Forbid
786A          movea.l (a7)+,a6            ; a6 wiederherstellen
786C          lea     128(a2),aØ          ; aØ -> ConUnit-Puffer
787Ø          bsr     FE6894              ; ---> dØ := Länge des Pufferinhalts
7874          tst.l   dØ                  ; Länge = Ø?
7876          beq.s   FE7884              ; ja: ->
7878          movea.l 14(a2),a1           ; a1 -> lh_Head (mp_MsgList)
787C          tst.l   (a1)                ; Liste leer?
787E          beq.s   FE7884              ; ja: ->
788Ø          bsr     FE73CØ              ; ---> cd_Read
7884 FE7884   move.l  a6,-(a7)            ; a6 retten
7886          movea.l 26(a6),a6           ; a6 := SysBase
788A          jsr     -8A(a6)             ; ---> Permit
788E          movea.l (a7)+,a6            ; a6 wiederherstellen
789Ø          movea.l a3,a1               ; a1 := a3 -> IORequest
7892          movem.l (a7)+,d2-d6/a2-a4   ; Register wiederherstellen
7896          moveq   #Ø,dØ               ; dØ := ok-Kode
7898          bra     FE5226              ; ---> IORequest beantworten
789C
789C ;------ Ausgabe-Abschluß
789C
789C FE789C   addq.l  #1,28(a3)           ; io_Data inkrementieren
78AØ          addq.l  #1,2Ø(a3)           ; io_Actual inkrementieren
78A4          subq.l  #1,24(a3)           ; io_Length dekrementieren
78A8          rts
78AA
78AA ;------ Tabelle von Routinenadressen
```

```
78AA
78AA FE78AA  DC.L    FE78BE              ; normal
78AE         DC.L    FE78E4              ; nach ESC ($1B)
78B2         DC.L    FE7932              ; nach ESC+SPACE
78B6         DC.L    FE797C              ; nach CSI ($9B)
78BA         DC.L    FE7A24              ; nach CSI+SPACE
78BE
78BE ;------ Normal
78BE
78BE FE78BE  move.w  d0,d1               ; d1 := d0 = Kode aus Text
78C0         btst.l  #7,d0               ; Kode >= $80?
78C4         bne.s   FE78D6              ; ja: ->
78C6         andi.w  #60,d1              ; $20 <= Kode < $80?
78CA         beq.s   FE78D2              ; ja: ->
78CC         movea.l 256(a2),a1          ; a1 := Funktionsvektor
78D0         jmp     (a1)                ; ---> Funktion aufrufen
78D2
78D2 FE78D2  bra     FE7B08              ; ---> CTRL-Kode bearbeiten
78D6
78D6 FE78D6  andi.w  #60,d1              ; $A0 <= Kode < $100?
78DA         beq.s   FE78E0              ; nein: ->
78DC         bra     FE7AF4              ; ---> Kode bearbeiten
78E0
78E0 FE78E0  bra     FE7C5C              ; ---> ESC-Sequenz bearbeiten
78E4
78E4 ;------ nach ESC
78E4
78E4 FE78E4  btst.l  #7,d0               ; Kode >= $80?
78E8         bne.s   FE78F6              ; ja: ->
78EA         btst.l  #6,d0               ; Kode >= $40?
78EE         bne.s   FE7914              ; ja: ->
78F0         btst.l  #5,d0               ; Kode >= $20?
78F4         bne.s   FE78FC              ; ja: ->
78F6 FE78F6  clr.w   2A0(a2)             ; Branch-Parameter := 'normal'
78FA         bra.s   FE78BE              ; ---> Normal
78FC
78FC FE78FC  btst.l  #4,d0               ; Kode >= $30?
7900         bne.s   FE7912              ; ja: ->
7902         lea     25A(a2),a0          ; a0 -> ESC-Puffer
7906         bsr     FE6814              ; ---> Byte in Puffer schreiben
```

```
790A          move.w  #2,2A0(a2)          ; Branch-Parameter := 'nach ESC+SPACE'
7910          bra.s   FE789C              ; ---> Abschluß
7912
7912 FE7912   bra.s   FE789C              ; ---> Abschluß
7914
7914 FE7914   bclr.l  #6,d0               ; Bit 6 löschen
7918          bclr.l  #5,d0               ; Bit 5 löschen
791C          bne.s   FE7926              ; Bit 5 war gesetzt: ->
791E          clr.w   2A0(a2)             ; Branch-Parameter := 'normal'
7922          bra     FE7C5C              ; ---> Abschluß
7926
7926 FE7926   bra     FE7CB4              ; ---> Kode bearbeiten
792A
792A ;------
792A
792A FE792A   movea.l 28(a3),a0           ; a0 := io_Data
792E          clr.w   d0                  ; d0 := 0
7930          move.b  (a0),d0             ; d0 := nächster Kode
7932
7932 ;------ nach ESC+SPACE
7932
7932 FE7932   btst.l  #7,d0               ; Kode >= $80?
7936          bne.s   FE78F6              ; ja: ->
7938          btst.l  #6,d0               ; Kode >= $40?
793C          bne.s   FE796C              ; ja: ->
793E          btst.l  #5,d0               ; Kode >= $20?
7942          beq.s   FE78F6              ; nein: ->
7944          btst.l  #4,d0               ; Kode >= $30?
7948          bne     FE7964              ; ja: ->
794C          lea     25A(a2),a0          ; a0 -> ESC-Puffer
7950          bsr     FE6814              ; ---> Byte in Puffer schreiben
7954          addq.l  #1,28(a3)           ; io_Data inkrementieren
7958          addq.l  #1,20(a3)           ; io_Actual inkrementieren
795C          subq.l  #1,24(a3)           ; io_Length dekrementieren
7960          bne.s   FE792A              ; noch nicht Null: ->
7962          rts
7964
7964 FE7964   bsr     FE789C              ; ---> Abschluß
7968          bra     FE7A10              ; ---> Branch-Parameter 'normal'
796C
```

```
796C FE796C  bsr     FE789C              ; ---> Abschluß
797Ø         bra     FE7A1Ø              ; ---> Branch-Parameter 'normal'
7974
7974 ;------
7974
7974 FE7974  movea.l 28(a3),aØ           ; aØ -> io_Data
7978         clr.w   dØ                  ; dØ := Ø
797A         move.b  (aØ),dØ             ; dØ := nächster Kode
797C
797C ;------ Nach CSI = $9B
797C
797C FE797C  btst.l  #7,dØ               ; Kode >= $8Ø?
798Ø         bne     FE78F6              ; ja: ->
7984         bclr.l  #6,dØ               ; Kode >= $6Ø?
7988         bne     FE7AØ6              ; ja: ->
798C         btst.l  #5,dØ               ; Kode >= $4Ø?
799Ø         beq     FE78F6              ; nein: ->
7994         btst.l  #4,dØ               ; Kode >= $5Ø?
7998         bne.s   FE79B2              ; ja: ->
799A         move.w  25A(a2),25C(a2)     ; ESC-Puffer löschen
79AØ         lea     25A(a2),aØ          ; aØ -> ESC-Puffer
79A4         bsr     FE6814              ; ---> Byte in Puffer schreiben
79A8         move.w  #4,2AØ(a2)          ; Branch-Parameter := 'CSI+SPACE'
79AE         bra     FE789C              ; ---> Abschluß
79B2
79B2 FE79B2  lea     26E(a2),aØ          ; aØ -> CSI-Puffer
79B6         adda.w  268(a2),aØ          ; aØ := aØ + BufferHead
79BA         cmpi.b  #39,dØ              ; Kode > '9'?
79BE         bgt.s   FE79EE              ; ja: ->
79CØ         move.w  dØ,d2               ; d2 := Kode
79C2         move.w  (aØ),dØ             ; dØ := nächster Kode
79C4         cmpi.w  #-5,dØ              ; dØ = -5, d.h. ';'?
79C8         bne.s   FE79CC              ; nein: ->
79CA         moveq   #Ø,dØ               ; dØ := Ø
79CC FE79CC  andi.w  #F,d2               ; Ziffernwert in d2 isolieren
79DØ         muls    #A,dØ               ; dØ := dØ * 1Ø
79D4         bmi.s   FE79EA              ; Ergebnis negativ: ->
79D6         add.w   d2,dØ               ; nächste Stelle addieren
79D8 FE79D8  move.w  dØ,(aØ)             ; und speichern
79DA FE79DA  addq.l  #1,28(a3)           ; io_Data inkrementieren
```

```
79DE          addq.l  #1,20(a3)           ; io_Actual inkrementieren
79E2          subq.l  #1,24(a3)           ; io_Length dekrementieren
79E6          bne.s   FE7974              ; noch nicht 0: ->
79E8          rts
79EA
79EA FE79EA   sub.w   d2,d0               ; nächste Stelle subtrahieren
79EC          bra.s   FE79D8              ; --->
79EE
79EE FE79EE   subi.w  #';',d0             ; d0 = ';'?
79F2          bne.s   FE7A00              ; nein: ->
79F4          moveq   #-5,d0              ; d0 := -5
79F6          lea     268(a2),a0          ; a0 -> CSI-Puffer
79FA          bsr     FE6834              ; ---> Wort in Puffer schreiben
79FE          bra.s   FE79DA              ; --->
7A00
7A00 FE7A00   bmi.s   FE79DA              ; d0 < ';': ->
7A02          neg.w   d0                  ; d0 negieren
7A04          bra.s   FE79D8              ; --->
7A06
7A06 FE7A06   cmpi.b  #2F,d0              ; d0 > 47?
7A0A          bhi.s   FE7A16              ; ja: ->
7A0C          bsr     FE7CC8              ; ---> CSI-Sequenz bearbeiten
7A10 FE7A10   clr.w   2A0(a2)             ; Branch-Parameter := 'normal'
7A14          rts
7A16
7A16 FE7A16   bsr     FE81EE              ; ---> CSI-Sequenz bearbeiten
7A1A          bra.s   FE7A10              ; --->
7A1C
7A1C ;------ Nach CSI+SPACE (Einsprung bei FE7A24)
7A1C
7A1C FE7A1C   movea.l 28(a3),a0           ; a0 -> io_Data
7A20          clr.w   d0                  ; d0 := 0
7A22          move.b  (a0),d0             ; d0 := nächster Kode
7A24 FE7A24   btst.l  #7,d0               ; Kode >= $80?
7A28          bne     FE78F6              ; ja: ->
7A2C          bclr.l  #6,d0               ; Kode >= $40?
7A30          bne.s   FE7A5A              ; ja: ->
7A32          btst.l  #5,d0               ; Kode >= $20?
7A36          beq     FE78F6              ; nein: ->
7A3A          btst.l  #4,d0               ; Kode >= $30?
```

```
7A3E          bne     FE78F6          ; ja: ->
7A42          lea     25A(a2),aØ      ; aØ -> ESC-Puffer
7A46          bsr     FE6814          ; ---> Byte in Puffer schreiben
7A4A          addq.l  #1,28(a3)       ; io_Data inkrementieren
7A4E          addq.l  #1,2Ø(a3)       ; io_Actual inkrementieren
7A52          subq.l  #1,24(a3)       ; io_Length dekrementieren
7A56          bne.s   FE7A1C          ; noch nicht Ø: ->
7A58          rts
7A5A
7A5A FE7A5A   cmpi.b  #2F,dØ          ; Kode > 47?
7A5E          bhi.s   FE7A66          ; ja: ->
7A6Ø          bsr     FE789C          ; ---> Abschluß
7A64          bra.s   FE7A1Ø          ; ---> Branch-Parameter 'normal'
7A66
7A66 FE7A66   bsr     FE82F2          ; ---> CSI-Sequenz bearbeiten
7A6A          bra.s   FE7A1Ø          ; ---> Branch-Parameter 'normal'
7A6C
7A6C ;------ Text ausgeben
7A6C
7A6C FE7A6C   movea.l aØ,a4           ; a4 := aØ -> Textpuffer
7A6E          move.l  24(a3),d1       ; d1 := io_Length
7A72 FE7A72   move.b  (aØ)+,dØ        ; dØ := nächster Kode
7A74          andi.b  #6Ø,dØ          ; Kode >= 96?
7A78          beq.s   FE7A7E          ; nein: ->
7A7A          subq.l  #1,d1           ; Zähler d1 dekrementieren
7A7C          bne.s   FE7A72          ; noch nicht Ø: Loop ->
7A7E FE7A7E   move.l  24(a3),d2       ; d2 := io_Length
7A82          sub.l   d1,d2           ; - Zahl der Kodes >= 96
7A84 FE7A84   move.w  26(a2),d1       ; d1 := cu_XCP
7A88          moveq   #Ø,d3           ; d3 := Ø
7A8A          move.w  2A(a2),d3       ; d3 := cu_XMax
7A8E          sub.w   d1,d3           ; - cu_XCP
7A9Ø          addq.w  #1,d3           ; + 1 = Restlänge der Zeile
7A92          cmp.w   d3,d2           ; Textlänge größer?
7A94          bge.s   FE7A98          ; ja: ->
7A96          move.w  d2,d3           ; d3 := Textlänge
7A98 FE7A98   moveq   #Ø,dØ           ; dØ := Ø
7A9A          move.w  d3,dØ           ; dØ := max(Textlänge,Zeilenrest)
7A9C          movea.l a4,aØ           ; aØ := a4 -> Textpuffer
7A9E          movea.l 22E(a2),a1      ; a1 -> wd_RPort
```

```
7AA2          move.w  26(a2),d1           ; d1 := cu_XCP
7AA6          mulu    2E(a2),d1           ; * cu_XRSize
7AAA          add.w   32(a2),d1           ; + cu_XROrigin
7AAE          move.w  d1,24(a1)           ; als rp_cp_x speichern
7AB2          move.w  28(a2),d1           ; d1 := cu_YCP
7AB6          mulu    3Ø(a2),d1           ; * cu_YRSize
7ABA          add.w   34(a2),d1           ; + cu_YROrigin
7ABE          add.w   3E(a1),d1           ; + rp_TxBaseLine
7AC2          move.w  d1,26(a1)           ; als rp_cp_y speichern
7AC6          bsr     FE6A32              ; ---> Kursor löschen
7ACA          move.l  a6,-(a7)            ; a6 retten
7ACC          movea.l 34(a6),a6           ; a6 := GfxBase
7ADØ          jsr     -3C(a6)             ; ---> Text
7AD4          movea.l (a7)+,a6            ; a6 wiederherstellen
7AD6          bsr     FE6A4E              ; ---> Kursor setzen
7ADA          add.l   d3,28(a3)           ; io_Data erhöhen
7ADE          add.l   d3,2Ø(a3)           ; io_Actual erhöhen
7AE2          sub.l   d3,24(a3)           ; io_Length erniedrigen
7AE6          move.w  d3,dØ               ; dØ := d3 = Länge des ausgeg. Textes
7AE8          bsr     FE6ACE              ; ---> Kursor dØ Spalten vorrücken
7AEC          adda.w  d3,a4               ; Zeiger in Textpuffer vorrücken
7AEE          sub.w   d3,d2               ; Textlänge vermindern
7AFØ          bhi.s   FE7A84              ; noch Text übrig: ->
7AF2          rts
7AF4
7AF4 ;------ Kodes von $AØ bis $FF
7AF4
7AF4 FE7AF4   moveq   #1,d2               ; Textlänge := 1
7AF6          move.b  (aØ),dØ             ; dØ := Kode aus Puffer
7AF8          ori.b   #8Ø,dØ              ; Bit 7 setzen
7AFC          move.w  dØ,-(a7)            ; Kode auf Stack legen
7AFE          movea.l a7,a4               ; a4 -> Kode im Stack
7BØØ          addq.w  #1,a4               ; + 1, weil Kode im L-Byte
7BØ2          bsr.s   FE7A84              ; ---> Zeichen ausgeben
7BØ4          addq.l  #2,a7               ; Stackzeiger korrigieren
7BØ6          rts
7BØ8
7BØ8 ;------ CTRL-Kodes Ø,...,$1F
7BØ8
7BØ8 FE7BØ8   add.b   dØ,dØ
```

```
7B0A          move.w   6(pc,d0.w),d0       ; d0 := Offset aus Tabelle
7B0E          jmp      2(pc,d0.w)          ; ---> Funktion aufrufen
7B12
7B12 FE7B12   DC.W     0194                ; FE7CA6   CTRL-@
7B14          DC.W     0194                ; FE7CA6        A
7B16          DC.W     0194                ; FE7CA6        B
7B18          DC.W     0194                ; FE7CA6        C
7B1A          DC.W     0194                ; FE7CA6        D
7B1C          DC.W     0194                ; FE7CA6        E
7B1E          DC.W     0194                ; FE7CA6        F
7B20          DC.W     0040                ; FE7B52        G  Beep
7B22          DC.W     005E                ; FE7B70        H  Backspace
7B24          DC.W     0068                ; FE7B7A        I  Tab
7B26          DC.W     0070                ; FE7B82        J  Line Feed
7B28          DC.W     0090                ; FE7BA2        K  VT = Cursor auf
7B2A          DC.W     009A                ; FE7BAC        L  FF = Clear Screen
7B2C          DC.W     00F4                ; FE7C06        M  CR = Carriage Return
7B2E          DC.W     0100                ; FE7C12        N  SO = Alt Char Set
7B30          DC.W     010C                ; FE7C1E        O  SI = Normal Char Set
7B32          DC.W     0194                ; FE7CA6        P
7B34          DC.W     0194                ; FE7CA6        Q
7B36          DC.W     0194                ; FE7CA6        R
7B38          DC.W     0194                ; FE7CA6        S
7B3A          DC.W     0194                ; FE7CA6        T
7B3C          DC.W     0194                ; FE7CA6        U
7B3E          DC.W     0194                ; FE7CA6        V
7B40          DC.W     0194                ; FE7CA6        W
7B42          DC.W     0194                ; FE7CA6        X
7B44          DC.W     0194                ; FE7CA6        Y
7B46          DC.W     0194                ; FE7CA6        Z
7B48          DC.W     0118                ; FE7C2A        [  ESC
7B4A          DC.W     0194                ; FE7CA6        \
7B4C          DC.W     0194                ; FE7CA6        ]
7B4E          DC.W     0194                ; FE7CA6        ^
7B50          DC.W     0194                ; FE7CA6        _
7B52
7B52 ;------ CTRL-G (Beep)
7B52
7B52 FE7B52   tst.l    38(a6)              ; Intuition geöffnet?
7B56          beq.s    FE7B6C              ; Nein: ->
```

```
7B58          movea.l 22(a2),aØ           ; aØ := cu_Window
7B5C          movea.l 2E(aØ),aØ           ; aØ := wd_WScreen
7B6Ø          move.l  a6,-(a7)            ; a6 retten
7B62          movea.l 38(a6),a6           ; a6 := IntuiBase
7B66          jsr     -6Ø(a6)             ; ---> DisplayBeep
7B6A          movea.l (a7)+,a6            ; a6 wiederherstellen
7B6C FE7B6C   bra     FE789C              ; ---> Abschluß
7B7Ø
7B7Ø ;------ CTRL-H (Backspace)
7B7Ø
7B7Ø FE7B7Ø   moveq   #1,dØ               ; dØ := 1
7B72          bsr     FE6BØA              ; ---> Kursor dØ Spalten zurück
7B76          bra     FE789C              ; ---> Abschluß
7B7A
7B7A ;------ CTRL-I (Tab)
7B7A
7B7A FE7B7A   bsr     FE7466              ; ---> Kursor zum nächsten TabStop
7B7E          bra     FE789C              ; ---> Abschluß
7B82
7B82 ;------ CTRL-J (Line Feed)
7B82
7B82 FE7B82   btst.b  #4,124(a2)          ; LineFeed-Newline-Flag gesetzt?
7B88          bne.s   FE7BF6              ; ja: ->
7B8A
7B8A ;------ ESC-D (IND, Kursor ab)
7B8A
7B8A FE7B8A   bclr.b  #Ø,2A4(a2)          ; Zeilenverkettungsflag löschen
7B9Ø          bne     FE789C              ; war gesetzt: Abschluß ->
7B94          moveq   #1,dØ               ; dØ := 1
7B96          bsr     FE6AA6              ; ---> Kursor dØ Zeilen abwärts
7B9A          bsr     FE7EC8              ; ---> Kursorposition in RastPort
7B9E          bra     FE789C              ; ---> Abschluß
7BA2
7BA2 ;------ CTRL-K (Kursor auf), ESC-M (RI, Kursor auf)
7BA2
7BA2 FE7BA2   moveq   #1,dØ               ; dØ := 1
7BA4          bsr     FE6A82              ; ---> Kursor dØ Zeilen aufwärts
7BA8          bra     FE789C              ; ---> Abschluß
7BAC
7BAC ;------ CTRL-L (FF = Clear Screen)
```

```
7BAC
7BAC FE7BAC  clr.l   26(a2)              ; cu_XCP := Ø
7BBØ         movea.l 22E(a2),a1          ; a1 := wd_RPort
7BB4         move.w  32(a2),d1           ; d1 := cu_XROrigin
7BB8         move.w  d1,24(a1)           ; rp_cp_x := d1
7BBC         move.w  34(a2),d1           ; d1 := cu_YROrigin
7BCØ         add.w   3E(a1),d1           ; + rp_TxBaseLine
7BC4         move.w  d1,26(a1)           ; rp_cp_y := d1
7BC8         bsr     FE6A32              ; ---> Kursor löschen
7BCC         bsr     FE68EC              ; ---> Display ab Kursor löschen
7BDØ         bsr     FE6A4E              ; ---> Kursor setzen
7BD4         bsr     FE6A66              ; ---> Kursor umsetzen
7BD8         bra     FE789C              ; ---> Abschluß
7BDC
7BDC ;------ CSI-F (Kursor in vorhergehende Zeile)
7BDC
7BDC FE7BDC  bsr     FE7D52              ; ---> dØ := Anzahl der Zeilen
7BEØ         bclr.b  #Ø,2A4(a2)          ; Zeilenverkettungsflag löschen
7BE6         bne     FE789C              ; war gesetzt: Abschluß ->
7BEA         bsr     FE6A82              ; ---> Kursor dØ Zeilen aufwärts
7BEE         bra.s   FE7CØ6              ; ---> CR
7BFØ
7BFØ ;------ CSI-E (Kursor in nachfolgende Zeile)
7BFØ
7BFØ FE7BFØ  bsr     FE7D52              ; ---> dØ := Anzahl der Zeilen
7BF4         bra.s   FE7BF8              ; --->
7BF6
7BF6 ;------ ESC-E (NEL = Next Line)
7BF6
7BF6 FE7BF6  moveq   #1,dØ               ; dØ := 1
7BF8 FE7BF8  bclr.b  #Ø,2A4(a2)          ; Zeilenverkettungsflag löschen
7BFE         bne     FE789C              ; war gesetzt: Abschluß ->
7CØ2         bsr     FE6AA6              ; ---> Kursor dØ Zeilen abwärts
7CØ6
7CØ6 ;------ CTRL-M (CR = Carriage Return)
7CØ6
7CØ6 FE7CØ6  clr.w   26(a2)              ; cu_XCP := Ø
7CØA         bsr     FE6A66              ; ---> Kursor umsetzen
7CØE         bra     FE789C              ; ---> Abschluß
7C12
```

```
7C12 ;------ CTRL-N (SO = Alt Character Set)
7C12
7C12 FE7C12  move.l  #FE7AF4,256(a2)      ; Ausgabevektor setzen
7C1A         bra     FE789C               ; ---> Abschluß
7C1E
7C1E ;------ CTRL-O (SI = Normal Character Set)
7C1E
7C1E FE7C1E  move.l  #FE7A6C,256(a2)      ; Ausgabevektor setzen
7C26         bra     FE789C               ; ---> Abschluß
7C2A
7C2A ;------ CTRL-[ (Escape)
7C2A
7C2A FE7C2A  move.w  #1,2A0(a2)           ; Branch-Parameter := 'nach ESC'
7C30         move.w  25A(a2),25C(a2)      ; ESC-Puffer löschen
7C36         bra     FE789C               ; ---> Abschluß
7C3A
7C3A ;------ ESC-H (HTS = Horizontal Tabulation Set)
7C3A
7C3A FE7C3A  bsr     FE74DE               ; ---> TabStop an Kursorposition setzen
7C3E         bra     FE789C               ; ---> Abschluß
7C42
7C42 ;------ CSI (Sequence beginnend mit $9B)
7C42
7C42 FE7C42  move.w  #3,2A0(a2)           ; Branch-Parameter := 'nach CSI'
7C48         move.w  268(a2),26A(a2)      ; CSI-Puffer löschen
7C4E         moveq   #FB,d0               ; d0 := -5
7C50         lea     268(a2),a0           ; a0 -> CSI-Puffer
7C54         bsr     FE6834               ; ---> Wort d0 in Puffer schreiben
7C58         bra     FE789C               ; ---> Abschluß
7C5C
7C5C ;------ ESC-Sequenz
7C5C
7C5C FE7C5C  add.b   d0,d0
7C5E         move.w  6(pc,d0.w),d0        ; d0 := Offset aus Tabelle
7C62         jmp     2(pc,d0.w)           ; ---> Funktion aufrufen
7C66
7C66 FE7C66  DC.W    0040                 ; FE7CA6    ESC-@
7C68         DC.W    0040                 ; FE7CA6        A
7C6A         DC.W    0040                 ; FE7CA6        B
7C6C         DC.W    0040                 ; FE7CA6        C
```

```
7C6E          DC.W    FF24                ; FE7B8A    ESC-D  IND = Kursor ab
7C7Ø          DC.W    FF9Ø                ; FE7BF6         E  NEL = Next Line
7C72          DC.W    ØØ4Ø                ; FE7CA6         F
7C74          DC.W    ØØ4Ø                ; FE7CA6         G
7C76          DC.W    FFD4                ; FE7C3A         H  HTS = Horiz Tab Set
7C78          DC.W    ØØ4Ø                ; FE7CA6         I
7C7A          DC.W    ØØ4Ø                ; FE7CA6         J
7C7C          DC.W    ØØ4Ø                ; FE7CA6         K
7C7E          DC.W    ØØ4Ø                ; FE7CA6         L
7C8Ø          DC.W    FF3C                ; FE7BA2         M  RI = Kursor auf
7C82          DC.W    ØØ4Ø                ; FE7CA6         N
7C84          DC.W    ØØ4Ø                ; FE7CA6         O
7C86          DC.W    ØØ4Ø                ; FE7CA6         P
7C88          DC.W    ØØ4Ø                ; FE7CA6         Q
7C8A          DC.W    ØØ4Ø                ; FE7CA6         R
7C8C          DC.W    ØØ4Ø                ; FE7CA6         S
7C8E          DC.W    ØØ4Ø                ; FE7CA6         T
7C9Ø          DC.W    ØØ4Ø                ; FE7CA6         U
7C92          DC.W    ØØ4Ø                ; FE7CA6         V
7C94          DC.W    ØØ4Ø                ; FE7CA6         W
7C96          DC.W    ØØ4Ø                ; FE7CA6         X
7C98          DC.W    ØØ4Ø                ; FE7CA6         Y
7C9A          DC.W    ØØ4Ø                ; FE7CA6         Z
7C9C          DC.W    FFDC                ; FE7C42         [  CSI = Control Sequ
7C9E          DC.W    ØØ4Ø                ; FE7CA6         \
7CAØ          DC.W    ØØ4Ø                ; FE7CA6         ]
7CA2          DC.W    ØØ4Ø                ; FE7CA6         ^
7CA4          DC.W    ØØ4Ø                ; FE7CA6         _
7CA6
7CA6 ;------ Abschluß
7CA6
7CA6 FE7CA6   addq.l  #1,28(a3)           ; io_Data inkrementieren
7CAA          addq.l  #1,2Ø(a3)           ; io_Actual inkrementieren
7CAE          subq.l  #1,24(a3)           ; io_Length dekrementieren
7CB2          rts
7CB4
7CB4 ;------ Nach ESC-Kleinbuchstabe
7CB4
7CB4 FE7CB4   cmpi.b  #3,dØ               ; ESC-c?
7CB8          beq.s   FE7CC2              ; ja: ->
```

```
7CBA FE7CBA  clr.w   2AØ(a2)            ; Branchparameter := 'normal'
7CBE         bra     FE789C            ; ---> Abschluß
7CC2
7CC2 FE7CC2  bsr     FE7524            ; ---> ConUnit und Window initialisieren
7CC6         bra.s   FE7CBA            ; ---> Branchparameter 'normal, Abschluß
7CC8
7CC8 ;------ Nach CSI (Control Sequence Introducer)
7CC8
7CC8 FE7CC8  add.b   dØ,dØ             ; Kode verdoppeln
7CCA         move.w  6(pc,dØ.w),dØ      ; dØ := Offset aus Tabelle
7CCE         jmp     2(pc,dØ.w)        ; ---> Funktion aufrufen
7CD2
7CD2 FE7CD2  DC.W    ØØE2              ; FE7DB4    CSI-@  Insert character
7CD4         DC.W    ØØ6C              ; FE7D3E        A  Cursor up
7CD6         DC.W    ØØ76              ; FE7D48        B  Cursor down
7CD8         DC.W    ØØ8C              ; FE7D5E        C  Cursor forward
7CDA         DC.W    ØØ96              ; FE7D68        D  Cursor backward
7CDC         DC.W    FF1E              ; FE7BFØ        E  Cursor next line
7CDE         DC.W    FFØA              ; FE7BDC        F  Cursor prec line
7CEØ         DC.W    FFD4              ; FE7CA6        G
7CE2         DC.W    ØØAØ              ; FE7D72        H  Cursor position
7CE4         DC.W    ØØDØ              ; FE7DA2        I  Cursor horiz tab
7CE6         DC.W    Ø1E4              ; FE7EB6        J  Erase to end of displ
7CE8         DC.W    Ø1D2              ; FE7EA4        K  Erase to end of line
7CEA         DC.W    Ø15E              ; FE7E3Ø        L  Insert line
7CEC         DC.W    Ø1B4              ; FE7E86        M  Delete line
7CEE         DC.W    FFD4              ; FE7CA6        N
7CFØ         DC.W    FFD4              ; FE7CA6        O
7CF2         DC.W    Ø14Ø              ; FE7E12        P  Delete character
7CF4         DC.W    FFD4              ; FE7CA6        Q
7CF6         DC.W    FFD4              ; FE7CA6        R
7CF8         DC.W    Ø22Ø              ; FE7EF2        S  Scroll up
7CFA         DC.W    Ø23E              ; FE7F1Ø        T  Scroll down
7CFC         DC.W    FFD4              ; FE7CA6        U
7CFE         DC.W    FFD4              ; FE7CA6        V
7DØØ         DC.W    Ø25E              ; FE7F3Ø        W  Cursor tab control
7DØ2         DC.W    FFD4              ; FE7CA6        X
7DØ4         DC.W    FFD4              ; FE7CA6        Y
7DØ6         DC.W    Ø294              ; FE7F66        Z  Cursor backward tab
7DØ8         DC.W    FFD4              ; FE7CA6        [
```

```
7D0A        DC.W    FFD4            ; FE7CA6    CSI-\
7D0C        DC.W    FFD4            ; FE7CA6        ]
7D0E        DC.W    FFD4            ; FE7CA6        ^
7D10        DC.W    FFD4            ; FE7CA6        _
7D12        DC.W    FFD4            ; FE7CA6
7D14        DC.W    FFD4            ; FE7CA6        a
7D16        DC.W    FFD4            ; FE7CA6        b
7D18        DC.W    FFD4            ; FE7CA6        c
7D1A        DC.W    FFD4            ; FE7CA6        d
7D1C        DC.W    FFD4            ; FE7CA6        e
7D1E        DC.W    00A0            ; FE7D72        f  Horiz & vert position
7D20        DC.W    02A8            ; FE7F7A        g  Tab clear
7D22        DC.W    02CC            ; FE7F9E        h  Set mode
7D24        DC.W    FFD4            ; FE7CA6        i
7D26        DC.W    FFD4            ; FE7CA6        j
7D28        DC.W    FFD4            ; FE7CA6        k
7D2A        DC.W    030E            ; FE7FE0        l  Reset mode
7D2C        DC.W    0320            ; FE7FF2        m  Select graphic rend
7D2E        DC.W    04B2            ; FE8184        n  Device status report
7D30        DC.W    FFD4            ; FE7CA6        o
7D32
```

```
7D32 ;------ Wort aus CSI-Puffer lesen
7D32
7D32 FE7D32  lea     268(a2),aØ          ; aØ -> CSI-Puffer
7D36         bsr     FE6876              ; ---> Wort aus Puffer lesen
7D3A         tst.l   dØ                  ; dØ = -1, wenn Puffer leer
7D3C         rts
7D3E
7D3E ;------ CSI-A (Kursor nach oben)
7D3E
7D3E FE7D3E  bsr.s   FE7D52              ; ---> dØ aus CSI-Puffer lesen
7D4Ø         bsr     FE6A82              ; ---> Kursor dØ Zeilen nach oben
7D44         bra     FE789C              ; ---> Abschluß
7D48
7D48 ;------ CSI-B (Kursor nach unten)
7D48
7D48 FE7D48  bsr.s   FE7D52              ; ---> dØ aus CSI-Puffer lesen
7D4A         bsr     FE6AA6              ; ---> Kursor dØ Zeilen nach unten
7D4E         bra     FE789C              ; ---> Abschluß
7D52
7D52 ;------ Wort aus CSI-Puffer lesen, Ersatzwert 1
7D52
7D52 FE7D52  bsr.s   FE7D32              ; ---> Wort aus CSI-Puffer lesen
7D54         bmi.s   FE7D5A              ; Puffer leer: ->
7D56         tst.w   dØ                  ; gelesenes Wort
7D58         bgt.s   FE7D5C              ; positiv: ->
7D5A FE7D5A  moveq   #1,dØ               ; dØ := 1 = Ersatzwert
7D5C FE7D5C  rts
7D5E
7D5E ;------ CSI-C (Kursor vorwärts)
7D5E
7D5E FE7D5E  bsr.s   FE7D52              ; ---> dØ aus CSI-Puffer lesen
7D6Ø         bsr     FE6ACE              ; ---> Kursor dØ Spalten vorwärts
7D64         bra     FE789C              ; ---> Abschluß
7D68
7D68 ;------ CSI-D (Kursor rückwärts)
7D68
7D68 FE7D68  bsr.s   FE7D52              ; ---> dØ aus CSI-Puffer lesen
7D6A         bsr     FE6BØA              ; ---> Kursor dØ Spalten zurück
7D6E         bra     FE789C              ; ---> Abschluß
7D72
```

```
7D72 ;------ CSI-H (Kursorposition)
7D72
7D72 FE7D72  bsr.s   FE7D52              ; ---> dØ := Zeile
7D74         cmp.w   2C(a2),dØ           ; dØ > cu_YMax?
7D78         bgt.s   FE7D96              ; ja: ->
7D7A         subq.w  #1,dØ               ; - 1
7D7C FE7D7C  move.w  dØ,28(a2)           ; cu_YCP := dØ
7D8Ø         bsr.s   FE7D52              ; ---> dØ := Spalte
7D82         cmp.w   2A(a2),dØ           ; dØ > cu_XMax?
7D86         bgt.s   FE7D9C              ; ja: ->
7D88         subq.w  #1,dØ               ; - 1
7D8A FE7D8A  move.w  dØ,26(a2)           ; cu_XCP := dØ
7D8E         bsr     FE6A66              ; ---> Kursor umsetzen
7D92         bra     FE789C              ; ---> Abschluß
7D96
7D96 FE7D96  move.w  2C(a2),dØ           ; dØ := cu_YMax
7D9A         bra.s   FE7D7C              ; --->
7D9C
7D9C FE7D9C  move.w  2A(a2),dØ           ; dØ := cu_XMax
7DAØ         bra.s   FE7D8A              ; --->
7DA2
7DA2 ;------ CSI-I (Kursor Horizontal Tab)
7DA2
7DA2 FE7DA2  bsr.s   FE7D52              ; ---> dØ aus CSI-Puffer lesen
7DA4         move.w  dØ,-(a7)            ; auf Stack
7DA6 FE7DA6  bsr     FE7466              ; ---> Kursor zum nächsten TabStop
7DAA         subq.w  #1,(a7)             ; Zahl auf Stack erniedrigen
7DAC         bne.s   FE7DA6              ; noch nicht Ø: wiederholen ->
7DAE         addq.w  #2,a7               ; Stackzeiger korrigieren
7DBØ         bra     FE789C              ; ---> Abschluß
7DB4
7DB4 ;------ CSI-@ (Zeichen einfügen)
7DB4
7DB4 FE7DB4  movem.l d2-d5,-(a7)         ; Register retten
7DB8         bsr.s   FE7DD4              ; ---> Scrollparameter bestimmen
7DBA         neg.l   dØ                  ; dx negieren
7DBC         move.l  a6,-(a7)            ; a6 retten
7DBE         movea.l 34(a6),a6           ; a6 := GfxBase
7DC2         jsr     -18C(a6)            ; ---> ScrollRaster
7DC6         movea.l (a7)+,a6            ; a6 wiederherstellen
```

```
7DC8          bsr      FE6A4E              ; ---> Kursor setzen
7DCC          movem.l  (a7)+,d2-d5         ; Register wiederherstellen
7DD0          bra      FE789C              ; ---> Abschluß
7DD4
7DD4 FE7DD4   bsr      FE7D52              ; ---> d0 aus CSI-Puffer lesen
7DD8          mulu     2E(a2),d0           ; * cu_XRSize = dx
7DDC          moveq    #0,d1               ; d1 := 0 = dy
7DDE          move.w   26(a2),d2           ; d2 := cu_XCP
7DE2          mulu     2E(a2),d2           ; * cu_XRSize
7DE6          add.w    32(a2),d2           ; + cu_XROrigin = xmin
7DEA          ext.l    d2                  ; d2 auf Langwort erweitern
7DEC          move.w   28(a2),d3           ; d3 := cu_YCP
7DF0          mulu     30(a2),d3           ; * cu_YRSize
7DF4          add.w    34(a2),d3           ; + cu_YROrigin = ymin
7DF8          ext.l    d3                  ; d3 auf Langwort erweitern
7DFA          move.w   36(a2),d4           ; d4 := cu_XRExtant = xmax
7DFE          ext.l    d4                  ; d4 auf Langwort erweitern
7E00          move.l   d3,d5               ; d5 := d3 = ymin
7E02          add.w    30(a2),d5           ; + cu_YRSize
7E06          subq.w   #1,d5               ; - 1 = ymax
7E08          bsr      FE6A32              ; ---> Kursor löschen
7E0C          movea.l  22E(a2),a1          ; a1 -> wd_RPort
7E10          rts
7E12
7E12 ;------ CSI-P (Zeichen löschen)
7E12
7E12 FE7E12   movem.l  d2-d5,-(a7)         ; Register retten
7E16          bsr.s    FE7DD4              ; ---> Scrollparameter bestimmen
7E18          move.l   a6,-(a7)            ; a6 retten
7E1A          movea.l  34(a6),a6           ; a6 := GfxBase
7E1E          jsr      -18C(a6)            ; ---> ScrollRaster
7E22          movea.l  (a7)+,a6            ; a6 wiederherstellen
7E24          bsr      FE6A4E              ; ---> Kursor setzen
7E28          movem.l  (a7)+,d2-d5         ; Register wiederherstellen
7E2C          bra      FE789C              ; ---> Abschluß
7E30
7E30 ;------ CSI-L (Zeile einfügen)
7E30
7E30 FE7E30   movem.l  d2-d5,-(a7)         ; Register retten
7E34          bsr.s    FE7E50              ; ---> Scrollparameter bestimmen
```

```
7E36          neg.l   d1                  ; dy negieren
7E38          move.l  a6,-(a7)            ; a6 retten
7E3A          movea.l 34(a6),a6           ; a6 := GfxBase
7E3E          jsr     -18C(a6)            ; ---> ScrollRaster
7E42          movea.l (a7)+,a6            ; a6 wiederherstellen
7E44          bsr     FE6A4E              ; ---> Kursor setzen
7E48          movem.l (a7)+,d2-d5         ; Register wiederherstellen
7E4C          bra     FE789C              ; ---> Abschluß
7E50
7E50 FE7E50   bsr     FE7D52              ; ---> d0 := Anzahl aus CSI-Puffer
7E54          move.l  d0,d1               ; d1 := d0 = Anzahl
7E56          moveq   #0,d0               ; d0 := 0 = dx
7E58          mulu    30(a2),d1           ; d1 := Anzahl * cu_YRSize = dy
7E5C          move.w  32(a2),d2           ; d2 := cu_XROrigin = xmin
7E60          ext.l   d2                  ; d2 auf Langwort erweitern
7E62          move.w  28(a2),d3           ; d3 := cu_YCP
7E66          mulu    30(a2),d3           ; * cu_YRSize
7E6A          add.w   34(a2),d3           ; + cu_YROrigin = ymin
7E6E          ext.l   d3                  ; d3 auf Langwort erweitern
7E70          move.w  36(a2),d4           ; d4 := cu_XRExtant = xmax
7E74          ext.l   d4                  ; d4 auf Langwort erweitern
7E76          move.w  38(a2),d5           ; d5 := cu_YRExtant = ymax
7E7A          ext.l   d5                  ; d5 auf Langwort erweitern
7E7C          bsr     FE6A32              ; ---> Kursor löschen
7E80          movea.l 22E(a2),a1          ; a1 := wd_RPort
7E84          rts
7E86
7E86 ;------ CSI-M (Zeile entfernen)
7E86
7E86 FE7E86   movem.l d2-d5,-(a7)         ; Register retten
7E8A          bsr.s   FE7E50              ; ---> Scrollparameter bestimmen
7E8C          move.l  a6,-(a7)            ; a6 retten
7E8E          movea.l 34(a6),a6           ; a6 := GfxBase
7E92          jsr     -18C(a6)            ; ---> ScrollRaster
7E96          movea.l (a7)+,a6            ; a6 wiederherstellen
7E98          bsr     FE6A4E              ; ---> Kursor setzen
7E9C          movem.l (a7)+,d2-d5         ; Register wiederherstellen
7EA0          bra     FE789C              ; ---> Abschluß
7EA4
```

```
7EA4 ;------ CSI-K (Zeile ab Kursor löschen)
7EA4
7EA4 FE7EA4  bsr.s   FE7EC8              ; ---> Kursorposition im RPort setzen
7EA6         bsr     FE6A32              ; ---> Kursor löschen
7EAA         bsr     FE68BØ              ; ---> Zeile ab Kursor löschen
7EAE         bsr     FE6A4E              ; ---> Kursor setzen
7EB2         bra     FE789C              ; ---> Abschluß
7EB6
7EB6 ;------ CSI-J (Display ab Kursor löschen)
7EB6
7EB6 FE7EB6  bsr.s   FE7EC8              ; ---> Kursorposition im RPort setzen
7EB8         bsr     FE6A32              ; ---> Kursor löschen
7EBC         bsr     FE68EC              ; ---> Display ab Kursor löschen
7ECØ         bsr     FE6A4E              ; ---> Kursor setzen
7EC4         bra     FE789C              ; ---> Abschluß
7EC8
7EC8 FE7EC8  movea.l 22E(a2),a1          ; a1 := wd_RPort
7ECC         move.w  26(a2),d1           ; d1 := cu_XCP
7EDØ         mulu    2E(a2),d1           ; * cu_XRSize
7ED4         add.w   32(a2),d1           ; + cu_XROrigin
7ED8         move.w  d1,24(a1)           ; rp_cp_x := d1
7EDC         move.w  28(a2),d1           ; d1 := cu_YCP
7EEØ         mulu    3Ø(a2),d1           ; * cu_YRSize
7EE4         add.w   34(a2),d1           ; + cu_YROrigin
7EE8         add.w   3E(a1),d1           ; + rp_TxBaseline
7EEC         move.w  d1,26(a1)           ; rp_cp_y := d1
7EFØ         rts
7EF2
7EF2 ;------ CSI-S (Scroll aufwärts)
7EF2
7EF2 FE7EF2  bsr     FE7D52              ; ---> dØ aus CSI-Puffer lesen
7EF6         move.w  dØ,d1               ; d1 := dØ = Dy
7EF8         ext.l   d1                  ; d1 auf Langwort erweitern
7EFA         moveq   #Ø,dØ               ; dØ := Ø = Dx
7EFC         movea.l 22E(a2),a1          ; a1 := wd_RPort
7FØØ         bsr     FE6A32              ; ---> Kursor löschen
7FØ4         bsr     FE6B3E              ; ---> Scroll Display
7FØ8         bsr     FE6A4E              ; ---> Kursor setzen
7FØC         bra     FE789C              ; ---> Abschluß
7F1Ø
```

```
7F1Ø ;------ CSI-T (Scroll abwärts)
7F1Ø
7F1Ø FE7F1Ø  bsr     FE7D52              ; ---> dØ aus CSI-Puffer lesen
7F14         move.w  dØ,d1               ; d1 := dØ = -Dy
7F16         neg.w   d1                  ; d1 := Dy
7F18         ext.l   d1                  ; d1 auf Langwort erweitern
7F1A         moveq   #Ø,dØ               ; dØ := Ø = Dx
7F1C         movea.l 22E(a2),a1          ; a1 := wd_RPort
7F2Ø         bsr     FE6A32              ; ---> Kursor löschen
7F24         bsr     FE6B3E              ; ---> Scroll Display
7F28         bsr     FE6A4E              ; ---> Kursor setzen
7F2C         bra     FE789C              ; ---> Abschluß
7F3Ø
7F3Ø ;------ CSI-W (Kursor Tab Control)
7F3Ø
7F3Ø FE7F3Ø  bsr     FE7D32              ; ---> dØ aus CSI-Puffer lesen
7F34         bmi     FE789C              ; Puffer leer: Fertig ->
7F38         tst.w   dØ                  ; dØ testen
7F3A         bmi.s   FE7F6Ø              ; negativ: TabStop setzen ->
7F3C         cmpi.w  #Ø,dØ
7F4Ø         beq.s   FE7F6Ø              ; = Ø: TabStop setzen ->
7F42         cmpi.w  #2,dØ
7F46         beq.s   FE7F5A              ; = 2: TabStop löschen ->
7F48         cmpi.w  #4,dØ
7F4C         beq.s   FE7F54              ; = 4: Alle TabStops löschen ->
7F4E         cmpi.w  #5,dØ
7F52         bne.s   FE7F3Ø              ; <> 5: Weiterlesen ->
7F54 FE7F54  bsr     FE751A              ; ---> Alle TabStops löschen
7F58         bra.s   FE7F3Ø              ; ---> weiterlesen
7F5A
7F5A FE7F5A  bsr     FE75ØØ              ; ---> TabStop an Kursorpos. löschen
7F5E         bra.s   FE7F3Ø              ; ---> weiterlesen
7F6Ø
7F6Ø FE7F6Ø  bsr     FE74DE              ; ---> TabStop an Kursorpos. setzen
7F64         bra.s   FE7F3Ø              ; ---> weiterlesen
7F66
7F66 ;------ CSI-Z (Kursor Tab rückwärts)
7F66
7F66 FE7F66  bsr     FE7D52              ; ---> dØ aus CSI-Puffer lesen
7F6A         move.w  dØ,-(a7)            ; und auf Stack legen
```

```
7F6C FE7F6C  bsr     FE74B0              ; ---> Einen TabStop zurück
7F70         subq.w  #1,(a7)             ; Anzahl dekrementieren
7F72         bne.s   FE7F6C              ; noch nicht 0: Loop ->
7F74         addq.w  #2,a7               ; Stackzeiger korrigieren
7F76         bra     FE789C              ; ---> Abschluß
7F7A
7F7A ;------ CSI-g (Tabmarken löschen)
7F7A
7F7A FE7F7A  bsr     FE7D32              ; ---> d0 aus CSI-Puffer lesen
7F7E         bmi     FE789C              ; Puffer leer: ->
7F82         tst.w   d0                  ; d0 testen
7F84         bmi.s   FE7F98              ; negativ: TabStop löschen ->
7F86         cmpi.w  #0,d0
7F8A         beq.s   FE7F98              ; = 0: TabStop löschen ->
7F8C         cmpi.w  #3,d0
7F90         bne.s   FE7F7A              ; <> 3: Weiterlesen
7F92         bsr     FE751A              ; ---> Alle TabStops löschen
7F96         bra.s   FE7F7A              ; ---> weiterlesen
7F98
7F98 FE7F98  bsr     FE7500              ; ---> TabStop an Kursorpos. löschen
7F9C         bra.s   FE7F7A              ; ---> weiterlesen
7F9E
7F9E ;------ CSI-h (Set Mode)
7F9E
7F9E FE7F9E  bsr     FE7D32              ; ---> d0 aus CSI-Puffer lesen
7FA2         bmi     FE789C              ; Puffer leer: Fertig ->
7FA6         bsr.s   FE7FB0              ; ---> d0 aufbereiten
7FA8         bmi.s   FE7F9E              ; Fehler: weiterlesen ->
7FAA         bset.b  d1,0(a0,d0.w)       ; ModeFlag setzen
7FAE         bra.s   FE7F9E              ; ---> weiterlesen
7FB0
7FB0 FE7FB0  tst.w   d0                  ; d0 testen
7FB2         bpl.s   FE7FC8              ; nicht negativ: ->
7FB4         cmpi.w  #-1F,d0             ; = -31 (entsteht aus '>1')?
7FB8         bne.s   FE7FBE              ; nein: ->
7FBA         moveq   #15,d0              ; d0 := 21 (AutoScroll)
7FBC         bra.s   FE7FCE              ; ---> auf Flagbit positionieren
7FBE
7FBE FE7FBE  cmpi.w  #-2F,d0             ; = -47 (entsteht aus '??')?
7FC2         bne.s   FE7FDC              ; nein: ->
```

```
7FC4          moveq   #16,d0              ; d0 := 22 (AutoWrap)
7FC6          bra.s   FE7FCE              ; ---> auf Flagbit positionieren
7FC8
7FC8 FE7FC8   cmpi.w  #14,d0              ; > 20?
7FCC          bgt.s   FE7FDC              ; ja: Unzulässig ->
7FCE FE7FCE   move.w  d0,d1               ; d1 := d0
7FD0          lsr.w   #3,d0               ; d0 := d0 \ 8
7FD2          andi.w  #7,d1               ; d1 := d1 mod 8
7FD6          lea     122(a2),a0          ; a0 -> ModeFlags
7FDA          rts
7FDC
7FDC FE7FDC   moveq   #FF,d0              ; d0 := -1
7FDE          rts
7FE0
7FE0 ;------ CSI-l (Reset Mode)
7FE0
7FE0 FE7FE0   bsr     FE7D32              ; ---> d0 aus CSI-Puffer lesen
7FE4          bmi     FE789C              ; Puffer leer: Fertig ->
7FE8          bsr.s   FE7FB0              ; ---> d0 aufbereiten
7FEA          bmi.s   FE7FE0              ; Fehler: weiterlesen ->
7FEC          bclr.b  d1,0(a0,d0.w)       ; ModeFlag löschen
7FF0          bra.s   FE7FE0              ; ---> weiterlesen
7FF2
7FF2 ;------ CSI-m (Select Graphic Rendition)
7FF2
7FF2 FE7FF2   bsr     FE7D32              ; ---> d0 aus CSI-Puffer lesen
7FF6          bmi     FE789C              ; Puffer leer: Fertig ->
7FFA          tst.w   d0                  ; d0 testen
7FFC          bpl.s   FE8000              ; nicht negativ: ->
7FFE          moveq   #0,d0               ; sonst durch 0 ersetzen
8000 FE8000   cmpi.w  #31,d0              ; d0 > 49?
8004          bgt.s   FE7FF2              ; ja: Weiterlesen ->
8006          add.b   d0,d0               ; d0 verdoppeln
8008          move.w  8(pc,d0.w),d0       ; d0 := Offset aus Tabelle
800C          jsr     4(pc,d0.w)          ; ---> Funktion aufrufen
8010          bra.s   FE7FF2              ; ---> weiterlesen
8012
8012 FE8012   DC.W    0066                ; FE8078    SGR 0  Plain
8014          DC.W    009E                ; FE80B0        1  Bold
8016          DC.W    00B4                ; FE80C6        2  Set FgPen = 2
```

```
8018        DC.W    00C8            ; FE80DA    SGR 3   Italic
801A        DC.W    00DE            ; FE80F0        4   Underscore
801C        DC.W    0064            ; FE8076        5
801E        DC.W    0064            ; FE8076        6
8020        DC.W    00F4            ; FE8106        7   Negative
8022        DC.W    0108            ; FE811A        8   Disable BitPlanes
8024        DC.W    0064            ; FE8076        9
8026        DC.W    0064            ; FE8076       10
8028        DC.W    0064            ; FE8076       11
802A        DC.W    0064            ; FE8076       12
802C        DC.W    0064            ; FE8076       13
802E        DC.W    0064            ; FE8076       14
8030        DC.W    0064            ; FE8076       15
8032        DC.W    0064            ; FE8076       16
8034        DC.W    0064            ; FE8076       17
8036        DC.W    0064            ; FE8076       18
8038        DC.W    0064            ; FE8076       19
803A        DC.W    0064            ; FE8076       20
803C        DC.W    0064            ; FE8076       21
803E        DC.W    0064            ; FE8076       22
8040        DC.W    0064            ; FE8076       23
8042        DC.W    0064            ; FE8076       24
8044        DC.W    0064            ; FE8076       25
8046        DC.W    0064            ; FE8076       26
8048        DC.W    0064            ; FE8076       27
804A        DC.W    0064            ; FE8076       28
804C        DC.W    0064            ; FE8076       29
804E        DC.W    0112            ; FE8124       30  CLR0
8050        DC.W    0116            ; FE8128       31  CLR1
8052        DC.W    011A            ; FE812C       32  CLR2
8054        DC.W    011E            ; FE8130       33  CLR3
8056        DC.W    0122            ; FE8134       34  CLR4
8058        DC.W    0126            ; FE8138       35  CLR5
805A        DC.W    012A            ; FE813C       36  CLR6
805C        DC.W    012E            ; FE8140       37  CLR7
805E        DC.W    0064            ; FE8076       38
8060        DC.W    0116            ; FE8128       39  CLR1
8062        DC.W    0142            ; FE8154       40  CLR0BG
8064        DC.W    0146            ; FE8158       41  CLR1BG
8066        DC.W    014A            ; FE815C       42  CLR2BG
```

```
8068          DC.W    014E                ; FE8160   SGR 43  CLR3BG
806A          DC.W    0152                ; FE8164       44  CLR4BG
806C          DC.W    0156                ; FE8168       45  CLR5BG
806E          DC.W    015A                ; FE816C       46  CLR6BG
8070          DC.W    015E                ; FE8170       47  CLR7BG
8072          DC.W    0064                ; FE8076       48
8074          DC.W    0142                ; FE8154       49  CLR0BG
8076
8076 FE8076   rts
8078
8078 ;------ SGR 0 (Normalschrift)
8078
8078 FE8078   movea.l 22E(a2),a1          ; a1 -> RastPort
807C          moveq   #FF,d0              ; alle Bits setzen
807E          move.b  d0,18(a1)           ; rp_Mask := d0
8082          movea.l 22E(a2),a1          ; a1 -> RastPort
8086          moveq   #1,d0               ; d0 := 1 = jam2
8088          move.l  a6,-(a7)            ; a6 retten
808A          movea.l 34(a6),a6           ; a6 := GfxBase
808E          jsr     -162(a6)            ; ---> SetDrMd
8092          movea.l (a7)+,a6            ; a6 wiederherstellen
8094          movea.l 22E(a2),a1          ; a1 -> RastPort
8098          moveq   #0,d0               ; d0 := 0 (normal)
809A          moveq   #7,d1               ; d1 := 7 = Enable
809C          move.l  a6,-(a7)            ; a6 retten
809E          movea.l 34(a6),a6           ; a6 := GfxBase
80A2          jsr     -5A(a6)             ; ---> SetSoftStyle
80A6          movea.l (a7)+,a6            ; a6 wiederherstellen
80A8          bsr.s   FE8128              ; ---> CLR1
80AA          bsr     FE8154              ; ---> CLR0BG
80AE          rts
80B0
80B0 ;------ SGR 1 (Fett)
80B0
80B0 FE80B0   movea.l 22E(a2),a1          ; a1 -> RastPort
80B4          moveq   #2,d0               ; d0 := 2 (bold)
80B6          moveq   #2,d1               ; d1 := 2 = Enable
80B8          move.l  a6,-(a7)            ; a6 retten
80BA          movea.l 34(a6),a6           ; a6 := GfxBase
80BE          jsr     -5A(a6)             ; ---> SetSoftStyle
```

```
80C2          movea.l (a7)+,a6            ; a6 wiederherstellen
80C4          rts
80C6
80C6 ;------ SGR 2 (Zeichenfarbe 2 setzen)
80C6
80C6 FE80C6   movea.l 22E(a2),a1          ; a1 -> RastPort
80CA          moveq   #2,d0               ; d0 := 2 (color)
80CC          move.l  a6,-(a7)            ; a6 retten
80CE          movea.l 34(a6),a6           ; a6 := GfxBase
80D2          jsr     -156(a6)            ; ---> SetAPen
80D6          movea.l (a7)+,a6            ; a6 wiederherstellen
80D8          rts
80DA
80DA ;------ SGR 3 (Kursiv)
80DA
80DA FE80DA   movea.l 22E(a2),a1          ; a1 -> RastPort
80DE          moveq   #4,d0               ; d0 := 4 (italic)
80E0          moveq   #4,d1               ; d1 := 4 = Enable
80E2          move.l  a6,-(a7)            ; a6 retten
80E4          movea.l 34(a6),a6           ; a6 := GfxBase
80E8          jsr     -5A(a6)             ; ---> SetSoftStyle
80EC          movea.l (a7)+,a6            ; a6 wiederherstellen
80EE          rts
80F0
80F0 ;------ SGR 4 (Unterstreichen)
80F0
80F0 FE80F0   movea.l 22E(a2),a1          ; a1 -> RastPort
80F4          moveq   #1,d0               ; d0 := 1 (underlined)
80F6          moveq   #1,d1               ; d1 := 1 = Enable
80F8          move.l  a6,-(a7)            ; a6 retten
80FA          movea.l 34(a6),a6           ; a6 := GfxBase
80FE          jsr     -5A(a6)             ; ---> SetSoftStyle
8102          movea.l (a7)+,a6            ; a6 wiederherstellen
8104          rts
8106
8106 ;------ SGR 7 (Invers)
8106
8106 FE8106   movea.l 22E(a2),a1          ; a1 -> RastPort
810A          moveq   #5,d0               ; d0 := 5 (jam2 & inversvid)
810C          move.l  a6,-(a7)            ; a6 retten
```

```
810E          movea.l 34(a6),a6           ; a6 := GfxBase
8112          jsr     -162(a6)            ; ---> SetDrMd
8116          movea.l (a7)+,a6            ; a6 wiederherstellen
8118          rts
811A
811A ;------ SGR 8 (Schreibsperre für alle Bitplanes)
811A
811A FE811A   movea.l 22E(a2),a1          ; a1 -> RastPort
811E          clr.b   18(a1)              ; rp_Mask löschen
8122          rts
8124
8124 ;------ SGR 30,...,37 (Vordergrundfarben 0,...,7)
8124
8124 FE8124   moveq   #0,d0
8126          bra.s   FE8142              ; --->
8128
8128 FE8128   moveq   #1,d0
812A          bra.s   FE8142              ; --->
812C
812C FE812C   moveq   #2,d0
812E          bra.s   FE8142              ; --->
8130
8130 FE8130   moveq   #3,d0
8132          bra.s   FE8142              ; --->
8134
8134 FE8134   moveq   #4,d0
8136          bra.s   FE8142              ; --->
8138
8138 FE8138   moveq   #5,d0
813A          bra.s   FE8142              ; --->
813C
813C FE813C   moveq   #6,d0
813E          bra.s   FE8142              ; --->
8140
8140 FE8140   moveq   #7,d0
8142 FE8142   movea.l 22E(a2),a1          ; a1 -> RastPort
8146          move.l  a6,-(a7)            ; a6 retten
8148          movea.l 34(a6),a6           ; a6 := GfxBase
814C          jsr     -156(a6)            ; ---> SetAPen
8150          movea.l (a7)+,a6            ; a6 wiederherstellen
```

```
8152          rts
8154
8154 ;------ SGR 40,...,47 (Hintergrundfarben 0,...,7)
8154
8154 FE8154   moveq   #0,d0
8156          bra.s   FE8172              ; --->
8158
8158 FE8158   moveq   #1,d0
815A          bra.s   FE8172              ; --->
815C
815C FE815C   moveq   #2,d0
815E          bra.s   FE8172              ; --->
8160
8160 FE8160   moveq   #3,d0
8162          bra.s   FE8172              ; --->
8164
8164 FE8164   moveq   #4,d0
8166          bra.s   FE8172              ; --->
8168
8168 FE8168   moveq   #5,d0
816A          bra.s   FE8172              ; --->
816C
816C FE816C   moveq   #6,d0
816E          bra.s   FE8172              ; --->
8170
8170 FE8170   moveq   #7,d0
8172 FE8172   movea.l 22E(a2),a1          ; a1 -> RastPort
8176          move.l  a6,-(a7)            ; a6 retten
8178          movea.l 34(a6),a6           ; a6 := GfxBase
817C          jsr     -15C(a6)            ; ---> SetBPen
8180          movea.l (a7)+,a6            ; a6 wiederherstellen
8182          rts
8184
8184 ;------ CSI-n (Device Status Report)
8184
8184 FE8184   bsr     FE7D32              ; ---> Wort aus CSI-Puffer lesen
8188          cmpi.l  #6,d0               ; d0 = 6?
818E          bne     FE789C              ; nein: Fertig ->
8192          move.l  a6,-(a7)            ; a6 retten
8194          movea.l 26(a6),a6           ; a6 := SysBase
```

```
8198          jsr     -84(a6)             ; ---> Forbid
819C          movea.l (a7)+,a6            ; a6 wiederherstellen
819E          moveq   #0,d0               ; d0 := 0
81A0          move.w  26(a2),d0           ; d0 := cu_XCP
81A4          addq.w  #1,d0               ; + 1
81A6          move.l  d0,-(a7)            ; auf Stack legen
81A8          move.w  28(a2),d0           ; d0 := cu_YCP
81AC          addq.w  #1,d0               ; + 1
81AE          move.l  d0,-(a7)            ; auf Stack legen
81B0          movea.l a7,a1               ; a1 := a7 -> Daten im Stack
81B2          movem.l a2-a3,-(a7)         ; Register retten
81B6          lea     2C(pc),a0           ; a0 := FE81E4 -> Formatstring
81BA          lea     128(a2),a3          ; a3 -> PutChData
81BE          lea     -1244(pc),a2        ; a2 := FE6F7C -> PutChProc
81C2          move.l  a6,-(a7)            ; a6 retten
81C4          movea.l 26(a6),a6           ; a6 := SysBase
81C8          jsr     -20A(a6)            ; ---> RawDoFmt
81CC          movea.l (a7)+,a6            ; a6 wiederherstellen
81CE          movem.l (a7)+,a2-a3         ; Register wiederherstellen
81D2          addq.w  #8,a7               ; Stackzeiger korrigieren
81D4          move.l  a6,-(a7)            ; a6 retten
81D6          movea.l 26(a6),a6           ; a6 := SysBase
81DA          jsr     -8A(a6)             ; ---> Permit
81DE          movea.l (a7)+,a6            ; a6 wiederherstellen
81E0          bra     FE789C              ; ---> Abschluß
81E4
81E4 FE81E4   DC.B    9B,'%ld;%ldR',0     ; Formatstring für Statusreport
81EE
81EE ;------ Sequenzen ab CSI-p
81EE
81EE FE81EE   subi.b  #30,d0              ; d0 := d0 - 48
81F2          add.b   d0,d0               ; verdoppeln
81F4          move.w  6(pc,d0.w),d0       ; d0 := Offset aus Tabelle
81F8          jmp     2(pc,d0.w)          ; ---> Funktion aufrufen
81FC
81FC FE81FC   DC.W    FAAA                ; FE7CA6    CSI-p
81FE          DC.W    FAAA                ; FE7CA6        q
8200          DC.W    FAAA                ; FE7CA6        r
8202          DC.W    FAAA                ; FE7CA6        s
8204          DC.W    0020                ; FE821C        t  Set Page Length
```

```
82Ø6          DC.W    ØØ42                ; FE823E    CSI-u  Set Line Length
82Ø8          DC.W    FAAA                ; FE7CA6         v
82ØA          DC.W    FAAA                ; FE7CA6         w
82ØC          DC.W    ØØ6E                ; FE826A         x  Set Left Offset
82ØE          DC.W    ØØ8E                ; FE828A         y  Set Top Offset
821Ø          DC.W    FAAA                ; FE7CA6         z
8212          DC.W    ØØAE                ; FE82AA         {  Set Raw Events
8214          DC.W    FAAA                ; FE7CA6         |
8216          DC.W    ØØD2                ; FE82CE         }  Reset Raw Events
8218          DC.W    FAAA                ; FE7CA6
821A          DC.W    FAAA                ; FE7CA6
821C
821C ;------ CSI-t (Set Page Length)
821C
821C FE821C   bsr     FE6A32              ; ---> Kursor löschen
822Ø          bsr     FE7D32              ; ---> dØ aus CSI-Puffer lesen
8224          tst.w   dØ                  ; dØ > Ø?
8226          ble.s   FE8236              ; nein: ->
8228          bset.b  #5,2A4(a2)          ; PageLength-Flag setzen
822E          subq.w  #1,dØ               ; Parameter dekrementieren
823Ø          move.w  dØ,2C(a2)           ; und als cu_YMax speichern
8234          bra.s   FE825E              ; --->
8236
8236 FE8236   bclr.b  #5,2A4(a2)          ; PageLength-Flag löschen
823C          bra.s   FE825E              ; --->
823E
823E ;------ CSI-u (Set Line Length)
823E
823E FE823E   bsr     FE6A32              ; ---> Kursor löschen
8242          bsr     FE7D32              ; ---> Wort aus CSI-Puffer lesen
8246          tst.w   dØ                  ; Wort gelesen?
8248          ble.s   FE8258              ; nein: ->
824A          bset.b  #4,2A4(a2)          ; LineLength-Flag setzen
825Ø FE824C   subq.w  #1,dØ               ; Parameter dekrementieren
8252 FE825Ø   move.w  dØ,2A(a2)           ; und als cu_XMax speichern
8256          bra.s   FE825E              ; --->
8258
8258 FE8258   bclr.b  #4,2A4(a2)          ; LineLength-Flag löschen
825E FE825E   bsr     FE76BØ              ; ---> Ausgabebereich initialisieren
8262          bsr     FE6A4E              ; ---> Kursor setzen
```

```
8266          bra     FE789C              ; ---> Abschluß
826A
826A ;------ CSI-x (Set Left Offset)
826A
826A FE826A   bsr     FE6A32              ; ---> Kursor löschen
826E          bsr     FE7D32              ; ---> Wort aus CSI-Puffer lesen
8272          tst.w   dØ                  ; dØ < Ø?
8274          bmi.s   FE8282              ; ja: ->
8276          bset.b  #6,2A4(a2)          ; LeftOffset-Flag setzen
827C          move.w  dØ,32(a2)           ; Parameter als cu_XROrigin speichern
828Ø          bra.s   FE825E              ; --->
8282
8282 FE8282   bclr.b  #6,2A4(a2)          ; LeftOffset-Flag löschen
8288          bra.s   FE825E              ; --->
828A
828A ;------ CSI-y (Set Top Offset)
828A
828A FE828A   bsr     FE6A32              ; ---> Kursor löschen
828E          bsr     FE7D32              ; ---> dØ aus CSI-Puffer lesen
8292          tst.w   dØ                  ; dØ < Ø?
8294          bmi.s   FE82A2              ; ja: ->
8296          bset.b  #7,2A4(a2)          ; TopOffset-Flag setzen
829C          move.w  dØ,34(a2)           ; Parameter als cu_YROrigin speichern
82AØ          bra.s   FE825E              ; --->
82A2
82A2 FE82A2   bclr.b  #7,2A4(a2)          ; TopOffset-Flag löschen
82A8          bra.s   FE825E              ; --->
82AA
82AA ;------ CSI-{ (Set Raw Events)
82AA
82AA FE82AA   bsr     FE7D32              ; ---> dØ aus CSI-Puffer lesen
82AE          bmi     FE789C              ; Puffer leer: Abschluß ->
82B2          tst.w   dØ                  ; dØ negativ?
82B4          bmi.s   FE82AA              ; ja: weiterlesen ->
82B6          cmpi.w  #12,dØ              ; Wort > ieClass_Max?
82BA          bgt.s   FE82AA              ; ja: weiterlesen ->
82BC          move.w  dØ,d1               ; d1 := dØ = ie_Class
82BE          lsr.w   #3,dØ               ; dØ := ie_Class \ 8
82CØ          andi.w  #7,d1               ; d1 := ie_Class mod 8
82C4          lea     125(a2),aØ          ; aØ -> cu_RawEvents
```

```
82C8          bset.b  d1,Ø(aØ,dØ.w)       ; RawEvent-Schalter setzen
82CC          bra.s   FE82AA              ; ---> weiterlesen
82CE
82CE ;------ CSI-} (Reset Raw Events)
82CE
82CE FE82CE   bsr     FE7D32              ; ---> dØ aus CSI-Puffer lesen
82D2          bmi     FE789C              ; Puffer leer: Abschluß ->
82D6          tst.w   dØ                  ; dØ negativ?
82D8          bmi.s   FE82CE              ; ja: weiterlesen ->
82DA          cmpi.w  #12,dØ              ; Wort > ieClass_Max?
82DE          bgt.s   FE82CE              ; ja: weiterlesen ->
82EØ          move.w  dØ,d1               ; d1 := dØ = ie_Class
82E2          lsr.w   #3,dØ               ; dØ := ie_Class \ 8
82E4          andi.w  #7,d1               ; d1 := ie_Class mod 8
82E8          lea     125(a2),aØ          ; aØ -> cu_RawEvents
82EC          bclr.b  d1,Ø(aØ,dØ.w)       ; RawEvent-Schalter rücksetzen
82FØ          bra.s   FE82CE              ; weiterlesen ->
82F2
82F2 ;------ CSI+SPACE $7Ø,...,$7F
82F2
82F2 FE82F2   move.l  dØ,-(a7)            ; dØ retten
82F4          lea     25A(a2),aØ          ; aØ -> ESC-Puffer
82F8          bsr     FE6854              ; ---> Byte aus Puffer lesen
82FC          cmpi.l  #' ',dØ             ; Space?
83Ø2          bne     FE8322              ; nein: Fertig ->
83Ø6          lea     25A(a2),aØ          ; aØ -> ESC-Puffer
83ØA          bsr     FE6894              ; ---> dØ := Zahl der Bytes im Puffer
83ØE          tst.l   dØ                  ; sind noch Bytes im Puffer?
831Ø          bne.s   FE8322              ; ja: Fertig ->
8312          move.l  (a7)+,dØ            ; dØ := CSI-Kode
8314          subi.b  #3Ø,dØ              ; - 48
8318          add.b   dØ,dØ               ; verdoppeln
831A          move.w  C(pc,dØ.w),dØ       ; dØ := Offset aus Tabelle
831E          jmp     8(pc,dØ.w)          ; ---> Funktion aufrufen
8322
8322 FE8322   addq.l  #4,a7               ; Stackzeiger korrigieren
8324          bra     FE789C              ; ---> Abschluß
8328
8328 FE8328   DC.W    ØØ2Ø                ; FE8348    CSI- p  Set Cursor Rendition
832A          DC.W    ØØ3E                ; FE8366    CSI- q  Window Status Requ
```

```
832C          DC.W    F97E                ; FE7CA6
832E          DC.W    F97E                ; FE7CA6
8330          DC.W    F97E                ; FE7CA6
8332          DC.W    F97E                ; FE7CA6
8334          DC.W    F97E                ; FE7CA6
8336          DC.W    F97E                ; FE7CA6
8338          DC.W    F97E                ; FE7CA6
833A          DC.W    F97E                ; FE7CA6
833C          DC.W    F97E                ; FE7CA6
833E          DC.W    F97E                ; FE7CA6
8340          DC.W    F97E                ; FE7CA6
8342          DC.W    F97E                ; FE7CA6
8344          DC.W    F97E                ; FE7CA6
8346          DC.W    F97E                ; FE7CA6
8348
8348 ;------ CSI- p (Set Cursor Rendition)
8348
8348 FE8348   bsr     FE6A32              ; ---> Kursor löschen
834C          bsr     FE7D32              ; ---> Wort aus CSI-Puffer lesen
8350          tst.w   d0                  ; Wort = 0?
8352          bne.s   FE835A              ; nein: ->
8354          bsr     FE6A20              ; ---> Kursor ausschalten
8358          bra.s   FE835E              ; ---> Kursor setzen, Abschluß
835A
835A FE835A   bsr     FE69A0              ; ---> Kursor einschalten
835E FE835E   bsr     FE6A4E              ; ---> Kursor setzen
8362          bra     FE789C              ; ---> Abschluß
8366
8366 ;------ CSI- q (Window Status Request)
8366
8366 FE8366   move.l  a6,-(a7)            ; a6 retten
8368          movea.l 26(a6),a6           ; a6 := SysBase
836C          jsr     -84(a6)             ; ---> Forbid
8370          movea.l (a7)+,a6            ; a6 wiederherstellen
8372          moveq   #0,d0               ; d0 := 0
8374          move.w  2A(a2),d0           ; d0 := cu_XMax
8378          addq.w  #1,d0               ; + 1
837A          move.l  d0,-(a7)            ; auf Stack legen
837C          move.w  2C(a2),d0           ; d0 := cu_YMax
8380          addq.w  #1,d0               ; + 1
```

```
8382          move.l  d0,-(a7)            ; auf Stack legen
8384          movea.l a7,a1               ; a1 -> Ausgabedaten
8386          movem.l a2-a3,-(a7)         ; Register retten
838A          lea     2C(pc),a0           ; a0 := FE83B8 -> Formatstring
838E          lea     128(a2),a3          ; a3 -> PutChData
8392          lea     -1418(pc),a2        ; a2 := FE6F7C -> PutChProc
8396          move.l  a6,-(a7)            ; a6 retten
8398          movea.l 26(a6),a6           ; a6 := SysBase
839C          jsr     -20A(a6)            ; ---> RawDoFmt
83A0          movea.l (a7)+,a6            ; a6 wiederherstellen
83A2          movem.l (a7)+,a2-a3         ; Register wiederherstellen
83A6          addq.w  #8,a7               ; Ausgabedaten löschen
83A8          move.l  a6,-(a7)            ; a6 retten
83AA          movea.l 26(a6),a6           ; a6 := SysBase
83AE          jsr     -8A(a6)             ; ---> Permit
83B2          movea.l (a7)+,a6            ; a6 wiederherstellen
83B4          bra     FE789C              ; ---> Abschluß
83B8
83B8 FE83B8   DC.B    9B,'1;1;%ld;%ld r',0,0       ; Formatstring
83C8
83C8 ************************************************************************
```

Keymap USA

```
83C8 *************** Keymap.resource ************************************
83C8
83C8 ;--------------------------------   Initialisierung
83C8
83C8 FE83C8  move.l  a2,-(a7)            ; a2 retten
83CA         moveq   #4A,d0              ; d0 := 74 Bytes
83CC         move.l  #10000,d1           ; d1 := Speichertyp CLEAR
83D2         jsr     -C6(a6)             ; ---> AllocMem für Struktur
83D6         tst.l   d0                  ; Speicher reserviert?
83D8         beq.s   FE841A              ; Nein: Fehler ->
83DA         movea.l d0,a2               ; a2 := d0 -> Reservierter Speicher
83DC         move.b  #8,8(a2)            ; Typ 8 (Resource) eintragen
83E2         move.l  #FE4FFE,A(a2)       ; Namenszeiger eintragen
83EA         lea     34(pc),a0           ; a0 := FE8420 -> Zeigertabelle
83EE         lea     1C(a2),a1           ; a1 -> KeyMap-Struktur
83F2         moveq   #16,d0              ; Zähler für 23 Worte
83F4 FE83F4  move.w  (a0)+,(a1)+         ; Zeiger in Struktur kopieren
83F6         dbra    d0,FE83F4           ; wiederholen bis d0 = -1 ->
83FA         lea     E(a2),a0            ; a0 -> List Header
83FE         lea     1C(a2),a1           ; a1 -> Knoten der KeyMap-Struktur
8402         move.l  a1,(a0)             ; List Header initialisieren
8404         clr.l   4(a0)               ; :
8408         move.l  a1,8(a0)            ; :
840C         move.l  a0,4(a1)            ; KeyMap-Knoten initialisieren
8410         addq.l  #4,a0               ; :
8412         move.l  a0,(a1)             ; :
8414         movea.l a2,a1               ; a1 -> Resource-Knoten
8416         jsr     -1E6(a6)            ; ---> AddResource
841A FE841A  movea.l (a7)+,a2            ; a2 wiederherstellen
841C         rts
841E
841E         DC.W    0
```

```
8420
8420 ;--------------------------------   Keymap-Struktur (wird ins RAM kopiert)
8420
8420 FE8420  DC.L    0                              ; ln_Succ
8424         DC.L    0                              ; ln_Pred
8428         DC.B    0                              ; ln_Type
8429         DC.B    0                              ; ln_Pri
842A         DC.L    FE887E                         ; ln_Name
842E
842E         DC.L    FE846C                         ; km_LowKeyMapTypes
8432         DC.L    FE84E4                         ; km_LowKeyMap
8436         DC.L    FE844E                         ; km_LowCapsable
843A         DC.L    FE845D                         ; km_LoRepeatable
843E         DC.L    FE84AC                         ; km_HiKeyMapTypes
8442         DC.L    FE85E4                         ; km_HiKeyMap
8446         DC.L    FE8456                         ; km_HiCapsable
844A         DC.L    FE8465                         ; km_HiRepeatable
844E
844E ;-------------------------   Keymap-Tabellen
844E
844E ;-----  LowCapsable
844E
844E FE844E  DC.B    0,0,FF,3,FF,1,FE,0
8456
8456 ;-----  HiCapsable
8456
8456 FE8456  DC.B    0,0,0,0,0,0,0,0
845D
845D ;-----  LoRepeatable
845D
845D FE845D  DC.B    FF,BF,FF,EF,FF,EF,FF,F7
8465
8465 ;-----  HiRepeatable
8465
8465 FE8465  DC.B    47,F4,FF,3,0,0,0
846C
846C ;-----  LowKeyMapTypes
846C
846C FE846C  DC.B    7,3,7,3,3,3,7,3,3,3,3,7,1,7,80,0
847C         DC.B    7,7,27,7,7,27,27,27,27,7,7,7,80,0,0,0
```

```
848C          DC.B    27,7,7,27,27,27,27,27,7,1,1,80,80,0,0,0
849C          DC.B    3,7,7,7,7,7,27,7,1,1,1,80,0,0,0,0
84AC
84AC ;-----   HiKeyMapTypes
84AC
84AC FE84AC   DC.B    22,0,41,0,4,2,0,80,80,80,0,80,41,41,41,41
84BC          DC.B    41,41,41,41,41,41,41,41,41,80,80,80,80,80,40
84CC          DC.B    80,80,80,80,80,80,80,80,80,80,80,80,80,80,80
84DC          DC.B    80,80,80,80,80,80,80,80
84E4
84E4 ;-----   LowKeyMap                       Ty C R Nr
84E4
84E4 FE84E4   DC.B    7E,60,7E,60           ;  7   1 00:  ~
84E8          DC.B    21,B9,21,31           ;  3   1 01:  1
84EC          DC.B    40,B2,40,32           ;  7   1 02:  2
84F0          DC.B    23,B3,23,33           ;  3   1 03:  3
84F4          DC.B    24,A2,24,34           ;  3   1 04:  4
84F8          DC.B    25,BC,25,35           ;  3   1 05:  5
84FC          DC.B    5E,BD,5E,36           ;  7   1 06:  6
8500          DC.B    26,BE,26,37           ;  3   1 07:  7
8504
8504          DC.B    2A,B7,2A,38           ;  3   1 08:  8
8508          DC.B    28,AB,28,39           ;  3   1 09:  9
850C          DC.B    29,BB,29,30           ;  3   1 0A:  0
8510          DC.B    5F,2D,5F,2D           ;  7   1 0B:  -
8514          DC.B    2B,3D,2B,3D           ;  1   1 0C:  =
8518          DC.B    7C,5C,7C,5C           ;  7   1 0D:  \
851C          DC.B    0,0,0,0               ; 80     0E:
8520          DC.B    0,0,0,30              ;  0   1 0F:  0  (Nummernblock)
8524
8524          DC.B    C5,E5,51,71           ;  7 1 1 10:  q
8528          DC.B    B0,B0,57,77           ;  7 1 1 11:  w
852C          DC.L    FE8730                ; 27 1 1 12:  e
8530          DC.B    AE,AE,52,72           ;  7 1 1 13:  r
8534          DC.B    DE,FE,54,74           ;  7 1 1 14:  t
8538          DC.L    FE87BC                ; 27 1 1 15:  y
```

```
853C        DC.L    FE87A0              ; 27 1 1 16:  u
8540        DC.L    FE874C              ; 27 1 1 17:  i
8544
8544        DC.L    FE8784              ; 27 1 1 18:  o
8548        DC.B    B6,B6,50,70         ;  7 1 1 19:  p
854C        DC.B    7B,5B,7B,5B         ;  7   1 1A:  [
8550        DC.B    7D,5D,7D,5D         ;  7   1 1B:  ]
8554        DC.B    0,0,0,0             ; 80     1C:
8558        DC.B    0,0,0,31            ;  0   1 1D:  1  (Nummernblock)
855C        DC.B    0,0,0,32            ;  0   1 1E:  2  (Nummernblock)
8560        DC.B    0,0,0,33            ;  0   1 1F:  3  (Nummernblock)
8564
8564        DC.L    FE8714                      ; 27 1 1 20:  a
8568        DC.B    A7,DF,53,73         ;  7 1 1 21:  s
856C        DC.B    D0,F0,44,64         ;  7 1 1 22:  d
8570        DC.L    FE86C4              ; 27 1 1 23:  f
8574        DC.L    FE86D4              ; 27 1 1 24:  g
8578        DC.L    FE86E4              ; 27 1 1 25:  h
857C        DC.L    FE86F4              ; 27 1 1 26:  j
8580        DC.L    FE8704              ; 27 1 1 27:  k
8584
8584        DC.B    A3,A3,4C,6C         ;  7 1 1 28:  l
8588        DC.B    3A,3B,3A,3B         ;  1   1 29:  ;
858C        DC.B    22,27,22,27         ;  1   1 2A:  '
8590        DC.B    0,0,0,0             ; 80     2B:
8594        DC.B    0,0,0,0             ; 80   1 2C:
8598        DC.B    0,0,0,34            ;  0   1 2D:  4  (Nummernblock)
859C        DC.B    0,0,0,35            ;  0   1 2E:  5  (Nummernblock)
85A0        DC.B    0,0,0,36            ;  0   1 2F:  6  (Nummernblock)
85A4
85A4        DC.B    BB,AB,3E,3C         ;  3   1 30:  <
85A8        DC.B    AC,B1,5A,7A         ;  7 1 1 31:  z
85AC        DC.B    F7,D7,58,78         ;  7 1 1 32:  x
85B0        DC.B    C7,E7,43,63         ;  7 1 1 33:  c
85B4        DC.B    AA,AA,56,76         ;  7 1 1 34:  v
85B8        DC.B    BA,BA,42,62         ;  7 1 1 35:  b
85BC        DC.L    FE8768              ; 27 1 1 36:  n
85C0        DC.B    BF,B8,4D,6D         ;  7 1 1 37:  m
85C4
85C4        DC.B    3C,2C,3C,2C         ;  1   1 38:  ,
```

```
85C8          DC.B    3E,2E,3E,2E          ;  1   1 39:  .
85CC          DC.B    3F,2F,3F,2F          ;  1   1 3A:  /
85D0          DC.B    0,0,0,0              ; 80     3B:
85D4          DC.B    0,0,0,2E             ;  0   1 3C:  .  (Nummernblock)
85D8          DC.B    0,0,0,37             ;  0   1 3D:  7  (Nummernblock)
85DC          DC.B    0,0,0,38             ;  0   1 3E:  8  (Nummernblock)
85E0          DC.B    0,0,0,39             ;  0   1 3F:  9  (Nummernblock)
85E4
85E4 ;-----   HiKeyMap
85E4
85E4 FE85E4   DC.L    FE87D8               ; 22   1 40:  Space
85E8          DC.B    0,0,0,8              ;  0   1 41:  Backspace
85EC          DC.L    FE87E2               ; 41   1 42:  Tab
85F0          DC.B    0,0,0,D              ;  0     43:  Enter (Nummernblock)
85F4          DC.B    0,0,A,D              ;  4     44:  Return
85F8          DC.B    0,0,9B,1B            ;  2     45:  Esc
85FC          DC.B    0,0,0,7F             ;  0   1 46:  Del
8600          DC.B    0,0,0,0              ; 80     47:
8604
8604          DC.B    0,0,0,0              ; 80     48:
8608          DC.B    0,0,0,0              ; 80     49:
860C          DC.B    0,0,0,2D             ;  0   1 4A:  -  (Nummernblock)
8610          DC.B    0,0,0,0              ; 80     4B:
8614          DC.L    FE87E9               ; 41   1 4C:  Kursor auf
8618          DC.L    FE87F1               ; 41   1 4D:  Kursor ab
861C          DC.L    FE87F9               ; 41   1 4E:  Kursor vor
8620          DC.L    FE8802               ; 41   1 4F:  Kursor zurück
8624
8624          DC.L    FE880B               ; 41   1 50:  F1
8628          DC.L    FE8816               ; 41   1 51:  F2
862C          DC.L    FE8821               ; 41   1 52:  F3
8630          DC.L    FE882C               ; 41   1 53:  F4
8634          DC.L    FE8837               ; 41   1 54:  F5
8638          DC.L    FE8842               ; 41   1 55:  F6
863C          DC.L    FE884D               ; 41   1 56:  F7
8640          DC.L    FE8858               ; 41   1 57:  F8
8644
8644          DC.L    FE8863               ; 41   1 58:  F9
8648          DC.L    FE886E               ; 41   1 59:  F10
864C          DC.B    0,0,0,0              ; 80     5A:
```

```
8650        DC.B    0,0,0,0             ; 80    5B:
8654        DC.B    0,0,0,0             ; 80    5C:
8658        DC.B    0,0,0,0             ; 80    5D:
865C        DC.B    0,0,0,0             ; 80    5E:
8660        DC.L    FE8879              ; 40    5F:  Help
8664
8664        DC.B    0,0,0,0             ; 80    60:  Shift links
8668        DC.B    0,0,0,0             ; 80    61:  Shift rechts
866C        DC.B    0,0,0,0             ; 80    62:  CapsLock
8670        DC.B    0,0,0,0             ; 80    63:  Ctrl
8674        DC.B    0,0,0,0             ; 80    64:  Alt links
8678        DC.B    0,0,0,0             ; 80    65:  Alt rechts
867C        DC.B    0,0,0,0             ; 80    66:  Amiga links
8680        DC.B    0,0,0,0             ; 80    67:  Amiga rechts
8684
8684        DC.B    0,0,0,0             ; 80    68:
8688        DC.B    0,0,0,0             ; 80    69:
868C        DC.B    0,0,0,0             ; 80    6A:
8690        DC.B    0,0,0,0             ; 80    6B:
8694        DC.B    0,0,0,0             ; 80    6C:
8698        DC.B    0,0,0,0             ; 80    6D:
869C        DC.B    0,0,0,0             ; 80    6E:
86A0        DC.B    0,0,0,0             ; 80    6F:
86A4
86A4        DC.B    0,0,0,0             ; 80    70:
86A8        DC.B    0,0,0,0             ; 80    71:
86AC        DC.B    0,0,0,0             ; 80    72:
86B0        DC.B    0,0,0,0             ; 80    73:
86B4        DC.B    0,0,0,0             ; 80    74:
86B8        DC.B    0,0,0,0             ; 80    75:
86BC        DC.B    0,0,0,0             ; 80    76:
86C0        DC.B    0,0,0,0             ; 80    77:
86C4
86C4 ;------ Dead Keys
86C4
86C4 FE86C4 DC.B    0,66,0,46,8,1,8,1,0,6,0,6,0,86,0,86      ; f
86D4
86D4 FE86D4 DC.B    0,67,0,47,8,2,8,2,0,7,0,7,0,87,0,87      ; g
86E4
86E4 FE86E4 DC.B    0,68,0,48,8,3,8,3,0,8,0,8,0,88,0,88      ; h
```

```
86F4
86F4 FE86F4  DC.B    Ø,6A,Ø,4A,8,4,8,4,Ø,A,Ø,A,Ø,8A,Ø,8A      ; j
87Ø4
87Ø4 FE87Ø4  DC.B    Ø,6B,Ø,4B,8,5,8,5,Ø,B,Ø,B,Ø,8B,Ø,8B      ; k
8714
8714 FE8714  DC.B    1,1Ø,1,16,Ø,E6,Ø,C6,Ø,1,Ø,1,Ø,81,Ø,81    ; a
8724         DC.B    61,E1,EØ,E2,E3,E4,41,C1,CØ,C2,C3,C4
873Ø
873Ø FE873Ø  DC.B    1,1Ø,1,16,Ø,A9,Ø,A9,Ø,5,Ø,5,Ø,85,Ø,85    ; e
874Ø         DC.B    65,E9,E8,EA,65,EB,45,C9,C8,CA,45,CB
874C
874C FE874C  DC.B    1,1Ø,1,16,Ø,A1,Ø,A6,Ø,9,Ø,9,Ø,89,Ø,89    ; i
875C         DC.B    69,ED,EC,EE,69,EF,49,CD,CC,CE,49,CF
8768
8768 FE8768  DC.B    1,1Ø,1,16,Ø,AD,Ø,AF,Ø,E,Ø,E,Ø,8E,Ø,8E    ; n
8778         DC.B    6E,6E,6E,6E,F1,6E,4E,4E,4E,4E,D1,4E
8784
8784 FE8784  DC.B    1,1Ø,1,16,Ø,F8,Ø,D8,Ø,F,Ø,F,Ø,8F,Ø,8F    ; o
8794         DC.B    6F,F3,F2,F4,F5,F6,4F,D3,D2,D4,D5,D6
87AØ
87AØ FE87AØ  DC.B    1,1Ø,1,16,Ø,B5,Ø,B5,Ø,15,Ø,15,Ø,95,Ø,95  ; u
87BØ         DC.B    75,FA,F9,FB,75,FC,55,DA,D9,DB,55,DC
87BC
87BC FE87BC  DC.B    1,1Ø,1,16,Ø,A4,Ø,A5,Ø,19,Ø,19,Ø,99,Ø,99  ; y
87CC         DC.B    79,FD,79,79,79,FF,59,DD,59,59,59,59
87D8
87D8 ;------ String Keys
87D8
87D8 FE87D8  DC.B    1,4, Ø,AØ, 2Ø, B4,6Ø,5E,7E,A8              ; Space
87E2
87E2 FE87E2  DC.B    1,4, 2,5, 9, 9B,5A                         ; Tab
87E9
87E9 FE87E9  DC.B    2,4, 2,6, 9B,41, 9B,54                     ; Kursor auf
87F1
87F1 FE87F1  DC.B    2,4, 2,6, 9B,42, 9B,53                     ; Kursor ab
87F9
87F9 FE87F9  DC.B    2,4, 3,6, 9B,43, 9B,2Ø,4Ø                  ; Kursor vor
88Ø2
88Ø2 FE88Ø2  DC.B    2,4, 3,6, 9B,44, 9B,2Ø,41                  ; Kursor zurück
88ØB
```

```
880B FE880B  DC.B    3,4, 4,7, 9B,30,7E, 9B,31,30,7E          ; F1
8816
8816 FE8816  DC.B    3,4, 4,7, 9B,31,7E, 9B,31,31,7E          ; F2
8821
8821 FE8821  DC.B    3,4, 4,7, 9B,32,7E, 9B,31,32,7E          ; F3
882C
882C FE882C  DC.B    3,4, 4,7, 9B,33,7E, 9B,31,33,7E          ; F4
8837
8837 FE8837  DC.B    3,4, 4,7, 9B,34,7E, 9B,31,34,7E          ; F5
8842
8842 FE8842  DC.B    3,4, 4,7, 9B,35,7E, 9B,31,35,7E          ; F6
884D
884D FE884D  DC.B    3,4, 4,7, 9B,36,7E, 9B,31,36,7E          ; F7
8858
8858 FE8858  DC.B    3,4, 4,7, 9B,37,7E, 9B,31,37,7E          ; F8
8863
8863 FE8863  DC.B    3,4, 4,7, 9B,38,7E, 9B,31,38,7E          ; F9
886E
886E FE886E  DC.B    3,4, 4,7, 9B,39,7E, 9B,31,39,7E          ; F10
8879
8879 FE8879  DC.B    3,2, 9B,3F,7E                            ; Help
887E
887E FE887E  DC.B    'usa',0
8882
8882         DC.W    0
8884
8884 ************************************************************************

→
```

Keymap D

```
0000 ***************  Keymap.d  ****************************************
0000
0000 ;-------------------------   Keymap-Struktur
0000
0000  KM0000   DC.L   KM003E      ; km_LowKeyMapTypes
0004           DC.L   KM00B6      ; km_LowKeyMap
0008           DC.L   KM0020      ; km_LowCapsable
000C           DC.L   KM002F      ; km_LowRepeatable
0010           DC.L   KM007E      ; km_HiKeyMapTypes
0014           DC.L   KM01B6      ; km_HiKeyMap
0018           DC.L   KM0028      ; km_HiCapsable
001C           DC.L   KM0037      ; km_HiRepeatable
0020
0020 ;-------------------------   Keymap-Tabellen
0020
0020 ;-------- km_LowCapsable
0020
0020  KM0020   DC.B   00,00,FF,07,FF,07,FE,00
0028
0028 ;-------- km_HiCapsable
0028
0028  KM0028   DC.B   00,00,00,00,00,00,00
002F
002F ;-------- km_LowRepeatable
002F
002F  KM002F   DC.B   FF,BF,FF,EF,FF,EF,FF,F7
0037
0037 ;-------- km_HiRepeatable
0037
0037  KM0037   DC.B   47,F4,FF,7F,00,00,00
003E
003E ;-------- km_LowKeyMapTypes
```

```
003E
003E  KM003E   DC.B   07,03,03,03,03,03,07,03,03,03,03,07,23,07,80,00
004E           DC.B   07,07,27,07,07,07,27,27,27,07,07,07,80,00,00,00
005E           DC.B   27,07,07,27,27,27,27,27,07,03,03,07,80,00,00,00
006E           DC.B   01,27,07,07,07,07,27,07,03,03,07,80,00,00,00,00
007E
007E ;-------- km_HiKeyMapTypes
007E
007E  KM007E   DC.B   22,00,41,00,04,02,00,80,80,80,00,80,41,41,41,41
008E           DC.B   41,41,41,41,41,41,41,41,41,41,05,05,00,00,00,40
009E           DC.B   80,80,80,80,80,80,80,80,80,80,80,80,80,80,80,80
00AE           DC.B   80,80,80,80,80,80,80,80
00B6
00B6 ;-------- km_LowKeyMap                Ty C R Nr
00B6
00B6  KM00B6   DC.B   7E,60,7E,60        ;  7   1 00:  '
00BA           DC.B   21,B9,21,31        ;  3   1 01:  1
00BE           DC.B   B2,40,22,32        ;  3   1 02:  2
00C2           DC.B   23,B3,A7,33        ;  3   1 03:  3
00C6           DC.B   A2,B0,24,34        ;  3   1 04:  4
00CA           DC.B   25,BC,25,35        ;  3   1 05:  5
00CE           DC.B   5E,BD,26,36        ;  7   1 06:  6
00D2           DC.B   26,BE,2F,37        ;  3   1 07:  7
00D6
00D6           DC.B   2A,B7,28,38        ;  3   1 08:  8
00DA           DC.B   28,AB,29,39        ;  3   1 09:  9
00DE           DC.B   29,BB,3D,30        ;  3   1 0A:  0
00E2           DC.B   5F,2D,3F,DF        ;  7   1 0B:  ß
00E6           DC.L   KM0296             ; 23   1 0C:  Akzent
00EA           DC.B   7C,5C,7C,5C        ;  7   1 0D:  \
00EE           DC.B   00,00,00,00        ; 80     0E:
00F2           DC.B   00,00,00,30        ;  0   1 0F:  0  (Num)
00F6
00F6           DC.B   C5,E5,51,71        ;  7 1 1 10:  q
00FA           DC.B   B0,B0,57,77        ;  7 1 1 11:  w
00FE           DC.L   KM02EE             ; 27 1 1 12:  e
0102           DC.B   AE,AE,52,72        ;  7 1 1 13:  r
0106           DC.B   DE,FE,54,74        ;  7 1 1 14:  t
010A           DC.B   A5,A4,5A,7A        ;  7 1 1 15:  z
010E           DC.L   KM0322             ; 27 1 1 16:  u
```

```
0112            DC.L    KM0356              ; 27 1 1 17:  i
0116
0116            DC.L    KM038A              ; 27 1 1 18:  o
011A            DC.B    B6,B6,50,70         ;  7 1 1 19:  p
011E            DC.B    7B,5B,DC,FC         ;  7 1 1 1A:  ü
0122            DC.B    7D,5D,2A,2B         ;  7   1 1B:  +
0126            DC.B    00,00,00,00         ; 80     1C:
012A            DC.B    00,00,00,31         ;  0   1 1D:  1  (Num)
012E            DC.B    00,00,00,32         ;  0   1 1E:  2  (Num)
0132            DC.B    00,00,00,33         ;  0   1 1F:  3  (Num)
0136
0136            DC.L    KM03BE              ; 27 1 1 20:  a
013A            DC.B    A7,DF,53,73         ;  7 1 1 21:  s
014E            DC.B    D0,F0,44,64         ;  7 1 1 22:  d
0142            DC.L    KM029E              ; 27 1 1 23:  f
0146            DC.L    KM02AE              ; 27 1 1 24:  g
014A            DC.L    KM02BE              ; 27 1 1 25:  h
015E            DC.L    KM02CE              ; 27 1 1 26:  j
0152            DC.L    KM02DE              ; 27 1 1 27:  k
0156
0156            DC.B    A3,A3,4C,6C         ;  7 1 1 28:  l
015A            DC.B    3A,3B,D6,F6         ;  3 1 1 29:  ö
016E            DC.B    22,27,C4,E4         ;  3 1 1 2A:  ä
0162            DC.B    5E,23,5E,23         ;  7   1 2B:  #
0166            DC.B    00,00,00,00         ; 80     2C:
016A            DC.B    00,00,00,34         ;  0   1 2D:  4  (Num)
017E            DC.B    00,00,00,35         ;  0   1 2E:  5  (Num)
0172            DC.B    00,00,00,36         ;  0   1 2F:  6  (Num)
0176
0176            DC.B    00,00,3E,3C         ;  1   1 30:  <
017A            DC.L    KM0426              ; 27 1 1 31:  y
018E            DC.B    F7,D7,58,78         ;  7 1 1 32:  x
0182            DC.B    C7,E7,43,63         ;  7 1 1 33:  c
0186            DC.B    AA,AA,56,76         ;  7 1 1 34:  v
018A            DC.B    BA,BA,42,62         ;  7 1 1 35:  b
```

```
019E            DC.L    KM03F2              ; 27 1 1 36:  n
0192            DC.B    BF,B8,4D,6D         ;  7 1 1 37:  m
0196
0196            DC.B    3C,2C,3B,2C         ;  3   1 38:  ,
019A            DC.B    3E,2E,3A,2E         ;  3   1 39:  .
01AE            DC.B    3F,2F,5F,2D         ;  7   1 3A:  -
01A2            DC.B    00,00,00,00         ; 80     3B:
01A6            DC.B    00,00,00,2E         ;  0   1 3C:  .  (Num)
01AA            DC.B    00,00,00,37         ;  0   1 3D:  7  (Num)
01BE            DC.B    00,00,00,38         ;  0   1 3E:  8  (Num)
01B2            DC.B    00,00,00,39         ;  0   1 3F:  9  (Num)
01B6
01B6 ;--------- H1KeyMap
01B6
01B6  KM01B6    DC.L    KM045A              ; 22   1 40:  Space
01BA            DC.B    00,00,00,08         ;  0   1 41:  Backspace
01BE            DC.L    KM0470              ; 41   1 42:  Tab
01C2            DC.B    00,00,00,0D         ;  0     43:  Enter
01C6            DC.B    00,00,0A,0D         ;  4     44:  Return
01CA            DC.B    00,00,9B,1B         ;  2   1 45:  Esc
01CE            DC.B    00,00,00,7F         ;  0     46:  Del
01D2            DC.B    00,00,00,00         ; 80     47:
01D6
01D6            DC.B    00,00,00,00         ; 80     48:
01DA            DC.B    00,00,00,00         ; 80     49:
01DE            DC.B    00,00,00,2D         ;  0   1 4A:  -  (Num)
01E2            DC.B    00,00,00,00         ; 80     4B:
01E6            DC.L    KM0477              ; 41   1 4C:  Kursor auf
01EA            DC.L    KM047F              ; 41   1 4D:  Kursor ab
01EE            DC.L    KM0487              ; 41   1 4E:  Kursor vor
01F2            DC.L    KM0490              ; 41   1 4F:  Kursor zurück
01F6
01F6            DC.L    KM0499              ; 41   1 50:  F1
01FA            DC.L    KM04A4              ; 41   1 51:  F2
01FE            DC.L    KM04AF              ; 41   1 52:  F3
0202            DC.L    KM04BA              ; 41   1 53:  F4
0206            DC.L    KM04C5              ; 41   1 54:  F5
020A            DC.L    KM04D0              ; 41   1 55:  F6
020E            DC.L    KM04DB              ; 41   1 56:  F7
0212            DC.L    KM04E6              ; 41   1 57:  F8
```

```
0216
0216            DC.L    KM04F1              ; 41   1 58:  F9
021A            DC.L    KM04FC              ; 41   1 59:  F10
021E            DC.B    1B,1B,7B,5B         ;  5   1 5A:  [  (Num)
0222            DC.B    1D,1D,7D,5D         ;  5   1 5B:  ]  (Num)
0226            DC.B    00,00,00,2F         ;  0   1 5C:  /  (Num)
022A            DC.B    00,00,00,2A         ;  0   1 5D:  *  (Num)
022E            DC.B    00,00,00,2B         ;  0   1 5E:  +  (Num)
0232            DC.L    KM0507              ; 40     5F:  Help
0236
0236            DC.B    00,00,00,00         ; 80     60:  Shift L
023A            DC.B    00,00,00,00         ; 80     61:  Shift R
023E            DC.B    00,00,00,00         ; 80     62:  CapsLock
0242            DC.B    00,00,00,00         ; 80     63:  Ctrl
0246            DC.B    00,00,00,00         ; 80     64:  Alt L
024A            DC.B    00,00,00,00         ; 80     65:  Alt R
024E            DC.B    00,00,00,00         ; 80     66:  Amiga L
0252            DC.B    00,00,00,00         ; 80     67:  Amiga R
0256
0256            DC.B    00,00,00,00         ; 80     68:
025A            DC.B    00,00,00,00         ; 80     69:
025E            DC.B    00,00,00,00         ; 80     6A:
0262            DC.B    00,00,00,00         ; 80     6B:
0266            DC.B    00,00,00,00         ; 80     6C:
026A            DC.B    00,00,00,00         ; 80     6D:
026E            DC.B    00,00,00,00         ; 80     6E:
0272            DC.B    00,00,00,00         ; 80     6F:
0276
0276            DC.B    00,00,00,00         ; 80     70:
027A            DC.B    00,00,00,00         ; 80     71:
027E            DC.B    00,00,00,00         ; 80     72:
0282            DC.B    00,00,00,00         ; 80     73:
0286            DC.B    00,00,00,00         ; 80     74:
028A            DC.B    00,00,00,00         ; 80     75:
028E            DC.B    00,00,00,00         ; 80     76:
0292            DC.B    00,00,00,00         ; 80     77:
0296
0296 ;-------- DeadKeys
0296
0296  KM0296    DC.B    08,61,08,62,00,3D,00,2B                          ; Akzent
```

```
029E
029E  KM029E   DC.B   00,66,00,46,08,61,08,61,00,06,00,06,00,86,00,86      ; f
02AE
02AE  KM02AE   DC.B   00,67,00,47,08,62,08,62,00,07,00,07,00,87,00,87      ; g
02BE
02BE  KM02BE   DC.B   00,68,00,48,08,03,08,03,00,08,00,08,00,88,00,88      ; h
02CE
02CE  KM02CE   DC.B   00,6A,00,4A,08,04,08,04,00,0A,00,0A,00,8A,00,8A      ; j
02DE
02DE  KM02DE   DC.B   00,6B,00,4B,08,05,08,05,00,0B,00,0B,00,8B,00,8B      ; k
02EE
02EE  KM02EE   DC.B   01,10,01,22,00,A9,00,A9,00,05,00,05,00,85,00,85      ; e
02FE           DC.B   65,E9,E8,EA,65,EB,E9,E9,EA,E9,E9,E9,E8,EA,E8,E8
030E           DC.B   E8,E8,45,C9,C8,CA,45,CB,C9,C9,CA,C9,C9,C9,C8,CA
031E           DC.B   C8,C8,C8,C8
0322
0322  KM0322   DC.B   01,10,01,22,00,B5,00,B5,00,15,00,15,00,95,00,95      ; u
0332           DC.B   75,FA,F9,FB,75,FC,FA,FA,FB,FA,FA,FA,F9,FB,F9,F9
0342           DC.B   F9,F9,55,DA,D9,DB,55,DC,DA,DA,DB,DA,DA,DA,D9,DB
0352           DC.B   D9,D9,D9,D9
0356
0356  KM0356   DC.B   01,10,01,22,00,A1,00,A6,00,09,00,09,00,89,00,89      ; i
0366           DC.B   69,ED,EC,EE,69,EF,ED,ED,EE,ED,ED,ED,EC,EE,EC,EC
0376           DC.B   EC,EC,49,CD,CC,CE,49,CF,CD,CD,CE,CD,CD,CD,CC,CE
0386           DC.B   CC,CC,CC,CC
038A
038A  KM038A   DC.B   01,10,01,22,00,F8,00,D8,00,0F,00,0F,00,8F,00,8F      ; o
039A           DC.B   6F,F3,F2,F4,F5,F6,F3,F3,F4,F3,F3,F3,F2,F4,F2,F2
03AA           DC.B   F2,F2,4F,D3,D2,D4,D5,D6,D3,D3,D4,D3,D3,D3,D2,D4
03BA           DC.B   D2,D2,D2,D2
03BE
03BE  KM03BE   DC.B   01,10,01,22,00,E6,00,C6,00,01,00,01,00,81,00,81      ; a
03CE           DC.B   61,E1,E0,E2,E3,E4,E1,E1,E2,E1,E1,E1,E0,E2,E0,E0
03DE           DC.B   E0,E0,41,C1,C0,C2,C3,C4,C1,C1,C2,C1,C1,C1,C0,C2
03EE           DC.B   C0,C0,C0,C0
03F2
03F2  KM03F2   DC.B   01,10,01,22,00,AD,00,AF,00,0E,00,0E,00,8E,00,8E      ; n
0402           DC.B   6E,6E,6E,6E,F1,6E,6E,6E,6E,6E,6E,6E,6E,6E,6E,6E
0412           DC.B   6E,6E,4E,4E,4E,4E,D1,4E,4E,4E,4E,4E,4E,4E,4E,4E
0422           DC.B   4E,4E,4E,4E
```

```
0426
0426  KM0426   DC.B   01,10,01,22,00,B1,00,AC,00,19,00,19,00,99,00,99      ; y
0436           DC.B   79,FD,79,79,79,FF,FD,FD,FD,FD,FD,79,79,79,79
0446           DC.B   79,79,59,DD,59,59,59,59,DD,DD,DD,DD,DD,DD,59,59
0456           DC.B   59,59,59,59
045A
045A ;-------- StringKeys
045A
045A  KM045A   DC.B   01,04,00,A0,20,B4,60,5E,7E,A8,B4,5E,B4,B4,B4,B4      ; Space
046A           DC.B   60,60,5E,60,60,60
0470
0470  KM0470   DC.B   01,04,02,05,09,9B,5A                ; Tab
0477
0477  KM0477   DC.B   02,04,02,06,9B,41,9B,54             ; Kursor auf
047F
047F  KM047F   DC.B   02,04,02,06,9B,42,9B,53             ; Kursor ab
0487
0487  KM0487   DC.B   02,04,03,06,9B,43,9B,20,40          ; Kursor vor
0490
0490  KM0490   DC.B   02,04,03,06,9B,44,9B,20,41          ; Kursor zurück
0499
0499  KM0499   DC.B   03,04,04,07,9B,30,7E,9B,31,30,7E    ; F1
04A4
04A4  KM04A4   DC.B   03,04,04,07,9B,31,7E,9B,31,31,7E    ; F2
04AF
04AF  KM04AF   DC.B   03,04,04,07,9B,32,7E,9B,31,32,7E    ; F3
04BA
04BA  KM04BA   DC.B   03,04,04,07,9B,33,7E,9B,31,33,7E    ; F4
04C5
04C5  KM04C5   DC.B   03,04,04,07,9B,34,7E,9B,31,34,7E    ; F5
04D0
04D0  KM04D0   DC.B   03,04,04,07,9B,35,7E,9B,31,35,7E    ; F6
04DB
04DB  KM04DB   DC.B   03,04,04,07,9B,36,7E,9B,31,36,7E    ; F7
04E6
04E6  KM04E6   DC.B   03,04,04,07,9B,37,7E,9B,31,37,7E    ; F8
04F1
04F1  KM04F1   DC.B   03,04,04,07,9B,38,7E,9B,31,38,7E    ; F9
04FC
04FC  KM04FC   DC.B   03,04,04,07,9B,39,7E,9B,31,39,7E    ; F10
```

```
0507
0507  KM0507   DC.B   03,02,9B,3F,7E                    ; Help
050C
050C ;-------------------------   Name
050C
050C  KM050C   DC.B   'd',0
050E
050E ************************************************************************
```

Timer device Datenbereich

```
-Ø36 *************** timer.device ****************************************
-Ø36
-Ø36          jmp     FE98BA              ; ---> CmpTime
-Ø3Ø          jmp     FE987A              ; ---> SubTime
-Ø2A          jmp     FE9896              ; ---> AddTime
-Ø24          jmp     FE9654              ; ---> AbortIO
-Ø1E          jmp     FE93C6              ; ---> BeginIO
-Ø18          jmp     FE93C2              ; ---> Extfunct
-Ø12          jmp     FE93C2              ; ---> Expunge
-ØØC          jmp     FE9472              ; ---> Close
-Ø06          jmp     FE944Ø              ; ---> Open
ØØØØ
ØØØØ ;--------------------------------   Library-Block
ØØØØ
ØØØØ          DC.L    2FDC                ; ln_Succ
ØØØ4          DC.L    2D54                ; ln_Pred
ØØØ8          DC.B    3                   ; ln_Type (3 = Device)
ØØØ9          DC.B    Ø                   ; ln_Pri
ØØØA          DC.L    FE91Ø6              ; ln_Name -> 'timer.device'
ØØØE
ØØØE          DC.B    4                   ; dev_Flags
ØØØF          DC.B    Ø                   ; dev_Pad
ØØ1Ø          DC.W    36                  ; dev_NegSize
ØØ12          DC.W    D6                  ; dev_PosSize
ØØ14          DC.W    21                  ; dev_Version
ØØ16          DC.W    36                  ; dev_Revision
ØØ18          DC.L    FE9114              ; dev_IdString
ØØ1C          DC.L    13C9ØØØØ            ; dev_LibSum
ØØ2Ø          DC.W    7                   ; dev_OpenCnt
ØØ22
```

```
0022 ;--------------------------------  Verschiedene Parameter
0022
0022          DC.W    32                ; = 50 = Bildwechselfrequenz
0024          DC.W    4E20              ; = 20000 = VBlank-Intervall in us
0026
0026          DC.L    4E20              ; = 20000 = VBlank-Intervall in us
002A
002A ;--------------------------------  Unit_MicroHz
002A
002A     00   DC.L    *002E             ; lh_Head       \ Request-Liste
002E     04   DC.L    0                 ; lh_Tail       |
0032     08   DC.L    *002A             ; lh_TailPred   |
0036     0C   DC.B    0                 ; lh_Type       |
0037     0D   DC.B    0                 ; lh_Pad        /
0038
0038     0E   DC.L    0                 ; nicht benutzt
003C     12   DC.L    3E8               ; Laufzeit bis Timer-Interrupt
0040     16   DC.W    0                 ; Unit-Nummer
0042     18   DC.W    2CB               ; Subtrahend für Zeitzähler
0044     1A   DC.L    1A80              ; ciaa.resource
0048     1E   DC.L    *0000             ; timer.device
004C     22   DC.L    FE97EC            ; Timer setzen und starten
0050     26   DC.L    FE9836            ; Timer stoppen
0054
0054 ;------ Interrupt
0054
0054     2A   DC.L    0                 ; ln_Succ
0058     2E   DC.L    0                 ; ln_Pred
005C     32   DC.B    2                 ; ln_Type (2 = Interrupt)
005D     33   DC.B    1E                ; ln_Pri = 30
005E     34   DC.L    FE9106            ; ln_Name -> 'timer.device'
0062     38   DC.L    *002A             ; is_Data -> Unit_MicroHz
0066     3C   DC.L    FE9726            ; is_Code -> Interrupt-Routine
006A
006A     40   DC.L    6096              ; lh_Head       \ Reply-Liste
006E     44   DC.L    0                 ; lh_Tail       |
0072     48   DC.L    *006A             ; lh_TailPred   |
0076     4C   DC.B    0                 ; lh_Type       |
0077     4D   DC.B    0                 ; lh_Pad        /
0078
```

```
0078 ;--------------------------------  Unit_VBlank
0078
0078     00  DC.L    60BE               ; lh_Head       \ Request-Liste
007C     04  DC.L    0                  ; lh_Tail       |
0080     08  DC.L    0F7C               ; lh_TailPred   |
0084     0C  DC.B    0                  ; lh_Type       |
0085     0D  DC.B    0                  ; lh_Pad        /
0086
0086     0E  DC.L    0                  ; nicht benutzt
008A     12  DC.L    4E20               ; Laufzeit bis Timer-Interrupt
008E     16  DC.W    1                  ; Unit-Nummer
0090     18  DC.W    0                  ; nicht benutzt
0092     1A  DC.L    0                  ; nicht benutzt
0096     1E  DC.L    *0000              ; timer.device
009A     22  DC.L    FE9834             ; Timer setzen und starten (hier: rts)
009E     26  DC.L    FE9878             ; Timer stoppen (hier: rts)
00A2
00A2 ;------ Interrupt
00A2
00A2     2A  DC.L    0902               ; ln_Succ
00A6     2E  DC.L    2E84               ; ln_Pred
00AA     32  DC.B    2                  ; ln_Type (2 = Interrupt)
00AB     33  DC.B    0                  ; ln_Pri
00AC     34  DC.L    FE9106             ; ln_Name -> 'timer.device'
00B0     38  DC.L    *0078              ; is_Data -> Unit_VBlank
00B4     3C  DC.L    FE96DA             ; is_Code -> Interrupt-Routine
00B8
00B8     40  DC.L    0                  ; lh_Head       \ Reply-Liste
00BC     44  DC.L    0                  ; lh_Tail       |
00C0     48  DC.L    *00B8              ; lh_TailPred   |
00C4     4C  DC.B    0                  ; lh_Type       |
00C5     4D  DC.B    0                  ; lh_Pad        /
00C6
00C6 ;-------------- Zeitbasis für Datum und Uhrzeit (gezählt ab 01.01.78)
00C6
00C6         DC.L    11CAA3A4           ; tv_Secs
00CA         DC.L    00038FF4           ; tv_Micro
00CE
```

```
00CE ;--------------------------------   Korrekturparameter, SysBase
00CE
00CE          DC.L    64A                 ; Korrektur wegen Eindeutigkeit
00D2
00D2          DC.L    676                 ; SysBase
00D6
00D6 ************************************************************************
```

Timer.device Routinen

```
90EC ***************** Timer.device-Modul ***********************************
90EC
90EC FE90EC  DC.W    4AFC                ; rt_Matchword
90EE         DC.L    FE90EC              ; rt_Matchtag
90F2         DC.L    FE98D8              ; rt_EndSkip
90F6         DC.B    1                   ; rt_Flags
90F7         DC.B    21                  ; rt_Version
90F8         DC.B    3                   ; rt_Type (3 = Device)
90F9         DC.B    32                  ; rt_Pri = 50
90FA         DC.L    FE9106              ; rt_Name
90FE         DC.L    FE9114              ; rt_IdString
9102         DC.L    FE9174              ; rt_Init
9106
9106 FE9106  DC.B    'timer.device',0,0
9114
9114 FE9114  DC.B    'timer 33.54 (31 Mar 1986)',0D,0A,0
9130
9130 FE9130  DC.B    'ciaa.resource',0
913E         DC.B    'timer.device',0,0
914C
914C ;--------------------------------  Vektoren für Sprungliste
914C
914C FE914C  DC.L    FE9440              ; Open
9150         DC.L    FE9472              ; Close
9154         DC.L    FE93C2              ; Expunge
9158         DC.L    FE93C2              ; Extfunct
915C         DC.L    FE93C6              ; BeginIO
9160         DC.L    FE9654              ; AbortIO
9164         DC.L    FE9896              ; AddTime
9168         DC.L    FE987A              ; SubTime
916C         DC.L    FE98BA              ; CmpTime
9170         DC.L    -1
```

```
9174
9174 ;--------------------------------   Initialisierung
9174
9174 FE9174  movem.l d2/a2-a3,-(a7)      ; Register retten
9178         lea     -4A(pc),a1          ; a1 := FE913Ø -> 'ciaa.resource'
917C         jsr     -1F2(a6)            ; ---> OpenResource
918Ø         tst.l   dØ                  ; Resource geöffnet?
9182         bne.s   FE919E              ; ja: ->
9184         movem.l d7/a5-a6,-(a7)      ; Register retten
9188         move.l  #15Ø58Ø2Ø,d7        ; d7 := Alert-Kode
918E         movea.l 4,a6                ; a6 := SysBase
9192         jsr     -6C(a6)             ; ---> Alert
9196         movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
919A         bra     FE93Ø8              ; ---> Ausgang
919E
919E FE919E  movea.l dØ,a3               ; a3 := dØ -> ciaa.resource
91AØ         lea     -56(pc),aØ          ; aØ := FE914C -> Vektortabelle
91A4         lea     FE9338,a1           ; a1 := FE9338 -> InitStruct-Tabelle
91AA         suba.l  a2,a2               ; a2 := Ø: Keine Initialisierung
91AC         move.l  #D6,dØ              ; dØ := 214 = Bytezahl für Library
91B2         jsr     -54(a6)             ; ---> MakeLibrary
91B6         tst.l   dØ                  ; Library ok?
91B8         bne.s   FE91D4              ; ja: ->
91BA         movem.l d7/a5-a6,-(a7)      ; Register retten
91BE         move.l  #15Ø2ØØØØ,d7        ; d7 := Alert-Kode
91C4         movea.l 4,a6                ; a6 := SysBase
91C8         jsr     -6C(a6)             ; ---> Alert
91CC         movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
91DØ         bra     FE93Ø8              ; ---> Ausgang
91D4
91D4 FE91D4  movea.l dØ,a2               ; a2 := dØ -> timer.device
91D6         move.l  a3,44(a2)           ; ciaa.resource eintragen
91DA         move.l  a6,D2(a2)           ; SysBase eintragen
91DE         move.l  a2,48(a2)           ; timer.device
91E2         move.l  a2,96(a2)           ; timer.device
91E6         lea     2A(a2),aØ           ; aØ -> Unit_MicroHz Request List Header
91EA         move.l  aØ,(aØ)             ; Macro NewList
91EC         addq.l  #4,(aØ)             ; :
91EE         clr.l   4(aØ)               ; :
91F2         move.l  aØ,8(aØ)            ; :
```

```
91F6          lea     78(a2),aØ          ; aØ -> Unit_VBlank Request List Header
91FA          move.l  aØ,(aØ)            ; Macro NewList
91FC          addq.l  #4,(aØ)            ; :
91FE          clr.l   4(aØ)              ; :
9202          move.l  aØ,8(aØ)           ; :
9206          lea     6A(a2),aØ          ; aØ -> Unit_MicroHz Reply List Header
920A          move.l  aØ,(aØ)            ; Macro NewList
920C          addq.l  #4,(aØ)            ; :
920E          clr.l   4(aØ)              ; :
9212          move.l  aØ,8(aØ)           ; :
9216          lea     B8(a2),aØ          ; aØ -> Unit_VBlank Reply List Header
921A          move.l  aØ,(aØ)            ; Macro NewList
921C          addq.l  #4,(aØ)            ; :
921E          clr.l   4(aØ)              ; :
9222          move.l  aØ,8(aØ)           ; :
9226          move.w  #1Ø,12Ø(a6)        ; Exec_Quantum := 16 = Task-Zeitquantum
922C          move.w  #1Ø,122(a6)        ; Exec_Elapsed := 16 = Task-Zeitzähler
9232          lea     A2(a2),a1          ; a1 -> Unit_VBlank Interrupt Struct
9236          lea     78(a2),aØ          ; aØ -> Unit_VBlank Data
923A          move.l  aØ,E(a1)           ; is_Data := aØ
923E          moveq   #5,dØ              ; dØ := 5 = Interrupt-Nummer
9240          jsr     -A8(a6)            ; ---> AddIntServer
9244          move.w  #8020,DFF09A       ; VBlank-Interrupt freigeben
924C          lea     54(a2),a1          ; a1 -> Unit_MicroHz Interrupt Struct
9250          lea     2A(a2),aØ          ; aØ -> Unit_MicroHz Data
9254          move.l  aØ,E(a1)           ; is_Data := aØ
9258          move.b  #1,dØ              ; dØ := 1 = Interrupt-Nummer
925C          move.l  a6,-(a7)           ; a6 retten
925E          movea.l 44(a2),a6          ; a6 -> ciaa.resource
9262          jsr     -6(a6)             ; ---> AddICRVector
9266          movea.l (a7)+,a6           ; a6 wiederherstellen
9268          moveq   #Ø,dØ              ; dØ := Ø
926A          move.b  212(a6),dØ         ; dØ := VBlankFrequency
926E          bne.s   FE9272             ; <> Ø: ->
9270          moveq   #3C,dØ             ; dØ := 6Ø
9272 FE9272   move.l  #ØF4240,d1         ; d1 := 1ØØØØØØ
9278          move.l  dØ,d2              ; d2 := dØ = VBlank-Frequenz
927A          divu    d2,d1              ; d1 := 1ØØØØØØ / VBlank-Frequenz
927C          moveq   #Ø,d2              ; d2 := Ø
927E          move.w  d1,d2              ; d2.w := d1 = 1ØØØØØØ / VBlank-Frequenz
```

```
9280          swap    d1                  ; d1.w := Divisionsrest
9282          lsr.l   #1,d0               ; d0 := Halbe VBlank-Frequenz
9284          cmp.w   d0,d1               ; Divisionsrest < halbe VBlank-Frequenz?
9286          blt.s   FE928A              ; ja: ->
9288          addq.l  #1,d2               ; d2 um 1 aufrunden
928A FE928A   move.l  d2,26(a2)           ; VBlank-Intervall in Mikrosekunden
928E          move.l  d2,8A(a2)           ; in timer.device eintragen
9292          bsr     FE953A              ; ---> Tageszeit-Uhr zurücksetzen
9296          move.b  #8,F00(a0)          ; Timer B One-Shot, Stop
929C          move.b  #FF,600(a0)         ; TBLO = Timer B low := -1
92A2          move.b  #FF,700(a0)         ; TBHI = Timer B high := -1, Timer Start
92A8          move.w  #4000,DFF09A        ; Interrupts sperren
92B0          addq.b  #1,126(a6)          ; IDNestCount erhöhen
92B4          bsr     FE930E              ; ---> Tageszeit-Uhr testen
92B8          move.b  #19,F00(a0)         ; Timer B Force-Load, One-Shot, Start
92BE          bsr     FE930E              ; ---> Tageszeit-Uhr Tickdauer messen
92C2          move.b  #8,F00(a0)          ; Timer B One-Shot, Stop
92C8          moveq   #0,d0               ; d0 := 0
92CA          move.b  700(a0),d0          ; d0.b := TBHI
92CE          lsl.l   #8,d0               ; um 8 Bits nach links schieben
92D0          move.b  600(a0),d0          ; d0.b := TBLO
92D4          subq.b  #1,126(a6)          ; IDNestCount erniedrigen
92D8          bge.s   FE92E2              ; Interrupts noch gesperrt: ->
92DA          move.w  #C000,DFF09A        ; Interrupts freigeben
92E2 FE92E2   cmpi.w  #CCDD,d0            ; d0 = (65535-Tickdauer) > 52445?
92E6          bhi.s   FE92F0              ; ja: ->
92E8          moveq   #32,d0              ; d0 := 50 = Netzfrequenz
92EA          move.w  #4E20,d1            ; d1 := 20000 = 1000000 / 50
92EE          bra.s   FE92F6              ; ---> Abschluß
92F0
92F0 FE92F0   moveq   #3C,d0              ; d0 := 60 = Netzfrequenz
92F2          move.w  #411B,d1            ; d1 := 16667 = 1000000 / 60
92F6 FE92F6   move.w  d0,22(a2)           ; VBlank-Frequenz eintragen
92FA          move.w  d1,24(a2)           ; VBlank-Intervall in us eintragen
92FE          move.b  d0,213(a6)          ; PowerSupplyFrequency eintragen
9302          movea.l a2,a1               ; a1 := a2 -> 'timer.device'
9304          jsr     -1B0(a6)            ; ---> AddDevice
9308 FE9308   movem.l (a7)+,d2/a2-a3      ; Register wiederherstellen
930C          rts
930E
```

```
930E ;------ Tageszeit-Uhr testen / Tickdauer messen
930E
930E FE930E  move.b  800(a0),d0          ; d0 := Event LSB
9312 FE9312  cmp.b   800(a0),d0          ; Event LSB verändert?
9316         bne.s   FE9336              ; ja: Fertig ->
9318         tst.b   700(a0)             ; TBHI = Timer B high
931C         bmi.s   FE9312              ; > 127: warten (maximal 45 ms) ->
931E         movem.l d7/a5-a6,-(a7)      ; Register retten
9322         move.l  #15000002,d7        ; d7 := Alert-Kode = AN_TMBadSupply
9328         movea.l 4,a6                ; a6 := SysBase
932C         jsr     -6C(a6)             ; ---> Alert
9330         movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
9334         bra.s   FE92F0              ; ---> Abschluß (Stackfehler!!!)
9336
9336 FE9336  rts
9338
9338 ;--------------------------------   Tabelle für InitStruct
9338
9338 FE9338  DC.W    E000                ; InitByte
933A         DC.W    8                   ; ln_Type
933C         DC.B    3                   ; = nt_Device
933D         DC.B    0
933E         DC.W    C000                ; InitLong
9340         DC.W    A                   ; ln_Name
9342         DC.L    FE9106              ; -> 'timer.device'
9346         DC.W    E000                ; InitByte
9348         DC.W    E                   ; lib_Flags
934A         DC.B    6                   ; = Changed, Sumused
934B         DC.B    0
934C         DC.W    D000                ; InitWord
934E         DC.W    14                  ; Version
9350         DC.W    21                  ; = 33
9352         DC.W    D000                ; InitWord
9354         DC.W    16                  ; Revision
9356         DC.W    36                  ; = 54
9358         DC.W    C000                ; InitLong
935A         DC.W    18                  ; IdString
935C         DC.L    FE9114              ; -> IdString
9360         DC.W    D000                ; InitWord
9362         DC.W    40                  ;
```

```
9364        DC.W    Ø               ; = Ø
9366        DC.W    CØØØ            ; InitLong
9368        DC.W    4C              ; MicroHz Timer setzen und starten
936A        DC.L    FE97EC          ; -> Timer B setzen und starten
936E        DC.W    CØØØ            ; InitLong
9370        DC.W    5Ø              ; MicroHz Timer stoppen
9372        DC.L    FE9836          ; -> Timer B Stop
9376        DC.W    EØØØ            ; InitByte
9378        DC.W    5C              ; ln_Type
937A        DC.B    2               ; = nt_Interrupt
937B        DC.B    Ø
937C        DC.W    EØØØ            ; InitByte
937E        DC.W    5D              ; ln_Pri
9380        DC.B    1E              ; = 30
9381        DC.B    Ø
9382        DC.W    CØØØ            ; InitLong
9384        DC.W    5E              ; ln_Name
9386        DC.L    FE91Ø6          ; -> 'timer.device'
938A        DC.W    CØØØ            ; InitLong
938C        DC.W    66              ; is_Code
938E        DC.L    FE9726          ; -> Interrupt-Routine Unit_MicroHz
9392        DC.W    DØØØ            ; InitWord
9394        DC.W    8E              ;
9396        DC.W    1               ; = 1
9398        DC.W    CØØØ            ; InitLong
939A        DC.W    9A              ; VBlank Timer setzen und starten
939C        DC.L    FE9834          ; -> rts
93AØ        DC.W    CØØØ            ; InitLong
93A2        DC.W    9E              ; VBlank Timer stoppen
93A4        DC.L    FE9878          ; -> rts
93A8        DC.W    EØØØ            ; InitByte
93AA        DC.W    AA              ; ln_Type
93AC        DC.B    2               ; = nt_Interrupt
93AD        DC.B    Ø
93AE        DC.W    CØØØ            ; InitLong
93BØ        DC.W    AC              ; ln_Name
93B2        DC.L    FE91Ø6          ; -> 'timer.device'
93B6        DC.W    CØØØ            ; InitLong
93B8        DC.W    B4              ; is_Code
93BA        DC.L    FE96DA          ; -> VBlank Interrupt Routine
```

```
93BE          DC.W    Ø
93CØ          DC.W    Ø
93C2
93C2 ;--------------------------------   Expunge, Extfunct
93C2
93C2 FE93C2   moveq   #Ø,dØ               ; dØ := Ø
93C4          rts
93C6
93C6 ;--------------------------------   BeginIO
93C6
93C6 FE93C6   move.b  #Ø,1F(a1)           ; io_Error := Ø
93CC          move.w  1C(a1),dØ           ; dØ := io_Command
93DØ          cmpi.w  #B,dØ               ; > 11?
93D4          bhi.s   FE93E2              ; ja: Fehler ->
93D6          move.b  #5,8(a1)            ; ln_Type := nt_Message
93DC          lsl.w   #2,dØ               ; dØ := Command * 4 = Tabellenoffset
93DE          jmp     E(pc,dØ.w)          ; ---> Routine aufrufen
93E2
93E2 FE93E2   move.b  #-3,1F(a1)          ; io_Error := ioerr_NoCmd
93E8          bsr     FE942A              ; ---> IORequest beantworten
93EC          rts
93EE
93EE ;------ Kommando-Verteiler
93EE
93EE FE93EE   bra     FE941E              ; ---> cmd_Invalid
93F2
93F2          bra     FE941E              ; ---> cmd_Reset
93F6
93F6          bra     FE941E              ; ---> cmd_Read
93FA
93FA          bra     FE941E              ; ---> cmd_Write
93FE
93FE          bra     FE941E              ; ---> cmd_Update
9402
9402          bra     FE941E              ; ---> cmd_Clear
9406
9406          bra     FE941E              ; ---> cmd_Stop
940A
940A          bra     FE941E              ; ---> cmd_Start
940E
```

```
940E          bra     FE941E              ; ---> cmd_Flush
9412
9412          bra     FE9550              ; ---> tr_AddRequest
9416
9416          bra     FE9484              ; ---> tr_GetSysTime
941A
941A          bra     FE9500              ; ---> tr_SetSysTime
941E
941E ;--------------------------------   Ungültiges Kommando
941E
941E FE941E   move.b  #-3,1F(a1)          ; io_Error := ioerr_NoCmd
9424          bsr     FE942A              ; ---> IORequest beantworten
9428          rts
942A
942A ;--------------------------------   IORequest beantworten
942A
942A FE942A   btst.b  #0,1E(a1)           ; io_Flags := QuickIO?
9430          bne.s   FE943E              ; ja: ->
9432          move.l  a6,-(a7)            ; a6 retten
9434          movea.l D2(a6),a6           ; a6 := SysBase
9438          jsr     -17A(a6)            ; ---> ReplyMsg
943C          movea.l (a7)+,a6            ; a6 wiederherstellen
943E FE943E   rts
9440
9440 ;--------------------------------   Open
9440
9440 FE9440   movem.l d2/a3,-(a7)         ; Register retten
9444          moveq   #0,d2               ; d2 := 0
9446          movea.l d0,a3               ; a3 := d0 = Unit
9448          cmpa.l  d2,a3               ; = 0 (Unit_MicroHz)?
944A          bne.s   FE9452              ; nein: ->
944C          lea     2A(a6),a3           ; a3 -> Unit_MicroHz
9450          bra.s   FE945C              ; --->
9452
9452 FE9452   addq.l  #1,d2               ; d2 := 1
9454          cmpa.l  d2,a3               ; a3 = 1 (Unit_VBlank)?
9456          bne.s   FE946A              ; nein: Fehler ->
9458          lea     78(a6),a3           ; a1 -> Unit_VBlank
945C FE945C   move.l  a3,18(a1)           ; Unit in io_Request eintragen
9460          addq.w  #1,20(a6)           ; Open-Zähler erhöhen
```

```
9464 FE9464  movem.l (a7)+,d2/a3          ; Register wiederherstellen
9468         rts
946A
946A FE946A  move.b  #-1,1F(a1)           ; io_Error := ioerr_OpenFail
947Ø         bra.s   FE9464               ; ---> Abschluß
9472
9472 ;-------------------------------   Close
9472
9472 FE9472  moveq   #-1,dØ
9474         move.l  dØ,14(a1)            ; io_Device := -1
9478         move.l  dØ,18(a1)            ; io_Unit := -1
947C         subq.w  #1,2Ø(a6)            ; Open-Zähler erniedrigen
948Ø         moveq   #Ø,dØ                ; dØ := Ø
9482         rts
9484
9484 ;-------------------------------   tr_GetSysTime
9484
9484 FE9484  move.l  a1,-(a7)             ; a1 retten
9486         lea     2Ø(a1),aØ            ; aØ -> tr_Secs
948A         bsr     FE9494               ; ---> Systemzeit übernehmen
948E         movea.l (a7)+,a1             ; a1 wiederherstellen
949Ø         bsr.s   FE942A               ; ---> IORequest beantworten
9492         rts
9494
9494 ;------ Systemzeit übernehmen
9494
9494 FE9494  movea.l 4,a1                 ; a1 := SysBase
9498         move.w  #4ØØØ,DFFØ9A         ; Interrupts sperren
94AØ         addq.b  #1,126(a1)           ; IDNestCnt erhöhen
94A4         move.l  C6(a6),(aØ)          ; tr_Secs := tv_Secs (Basiswert)
94A8         move.l  CA(a6),4(aØ)         ; tr_Micro := tv_Micro (Basiswert)
94AE         moveq   #Ø,dØ                ; dØ := Ø
94BØ         lea     BFEØØ1,a1            ; a1 -> ciaa
94B6         move.b  AØØ(a1),dØ           ; dØ.b := Event MSB
94BA         lsl.l   #8,dØ                ; um 8 Bits nach links schieben
94BC         move.b  9ØØ(a1),dØ           ; dØ.b := Event 8-15
94CØ         lsl.l   #8,dØ                ; um 8 Bits nach links schieben
94C2         move.b  8ØØ(a1),dØ           ; dØ.b := Event LSB
94C6         movea.l 4,a1                 ; a1 := SysBase
94CA         subq.b  #1,126(a1)           ; IDNestCnt erniedrigen
```

```
94CE          bge.s   FE94D8              ; Interrupts noch gesperrt: ->
94DØ          move.w  #CØØØ,DFFØ9A        ; Interrupts freigeben
94D8 FE94D8   divu    22(a6),dØ           ; dØ durch VBlank-Frequenz dividieren
94DC          swap    dØ                  ; dØ.w := Divisionsrest
94DE          move.w  dØ,d1               ; d1 := dØ.w = Divisionsrest
94EØ          mulu    24(a6),d1           ; mal Mikrosekunden / VBlank-Frequenz
94E4          add.l   CE(a6),d1           ; + Korrektur wegen Eindeutigkeit
94E8          move.l  d1,-(a7)            ; Mikrosekunden auf Stack
94EA          swap    dØ                  ; dØ.w = Sekunden
94EC          movea.w dØ,a1               ; a1 := dØ = Sekunden
94EE          move.l  a1,-(a7)            ; Sekunden auf Stack
94FØ          movea.l a7,a1               ; a1 := a7 -> Zeitdaten im Stack
94F2          bsr     FE9896              ; ---> AddTime
94F6          addq.l  #8,a7               ; Stackzeiger korrigieren
94F8          moveq   #1,dØ               ; dØ := 1
94FA          add.l   dØ,CE(a6)           ; Korrekturwert erhöhen
94FE          rts
95ØØ
95ØØ ;--------------------------------   tr_SetSysTime
95ØØ
95ØØ FE95ØØ   movea.l 4,aØ               ; aØ := SysBase
95Ø4          move.w  #4ØØØ,DFFØ9A        ; Interrupts sperren
95ØC          addq.b  #1,126(aØ)          ; IDNestCnt erhöhen
951Ø          move.l  2Ø(a1),C6(a6)       ; tv_Secs := tr_Secs
9516          move.l  24(a1),CA(a6)       ; tv_Micro := tr_Micro
951C          bsr.s   FE9536              ; ---> Tageszeit-Uhr zurücksetzen
951E          movea.l 4,aØ                ; aØ := SysBase
9522          subq.b  #1,126(aØ)          ; IDNestCnt erniedrigen
9526          bge.s   FE953Ø              ; Interrupts noch gesperrt: ->
9528          move.w  #CØØØ,DFFØ9A        ; Interrupts freigeben
953Ø FE953Ø   bsr     FE942A              ; ---> IORequest beantworten
9534          rts
9536
9536 ;------ Tageszeit-Uhr und Korrekturwert zurücksetzen
9536
9536 FE9536   clr.l   CE(a6)              ; Korrekturwert löschen
953A FE953A   lea     BFEØØ1,aØ           ; aØ -> ciaa.pra
954Ø          moveq   #Ø,dØ               ; dØ := Ø
9542          move.b  dØ,AØØ(aØ)          ; Event MSB  := Ø
9546          move.b  dØ,9ØØ(aØ)          ; Event 8-15 := Ø
```

```
954A          move.b  dØ,8ØØ(aØ)          ; Event LSB  := Ø
954E          rts
9550
9550 ;--------------------------------   tr_AddRequest
9550
9550 FE9550   move.l  a3,-(a7)            ; a3 retten
9552          bclr.b  #Ø,1E(a1)           ; io_Flags.QuickIO löschen
9558          movea.l 4,aØ                ; aØ := SysBase
955C          move.w  #4ØØØ,DFFØ9A        ; Interrupts sperren
9564          addq.b  #1,126(aØ)          ; IDNestCnt erhöhen
9568          movea.l 18(a1),a3           ; a3 := io_Unit
956C          cmpa.l  8(a3),a3            ; Request-Liste leer?
9570          bne.s   FE958A              ; nein: Neuen Request einordnen ->
9572          move.l  (a3),dØ             ; Macro AddHead
9574          move.l  a1,(a3)             ; :
9576          movem.l dØ/a3,(a1)          ; :
957A          movea.l dØ,aØ               ; :
957C          move.l  a1,4(aØ)            ; :
958Ø          movea.l 22(a3),aØ           ; aØ -> Timerroutine
9584          jsr     (aØ)                ; ---> Timer setzen und starten
9586          bra     FE963E              ; ---> Abschluß
958A
958A ;------ Neuen Request in Request-Liste einordnen
958A
958A FE958A   movem.l a2/a4-a5,-(a7)      ; Register retten
958E          suba.l  a2,a2               ; a2 := Ø
9590          movea.l a1,a5               ; a5 := a1 -> IORequest
9592          movea.l (a3),a4             ; a4 -> 1. Request in der Liste
9594          move.l  2Ø(a4),dØ           ; dØ := tr_Secs aus Liste
9598          cmp.l   2Ø(a5),dØ           ; = tr_Secs aus IORequest?
959C          bne.s   FE95A6              ; nein: ->
959E          move.l  24(a4),dØ           ; dØ := tr_Micro aus Liste
95A2          cmp.l   24(a5),dØ           ; = tr_Micro aus IORequest?
95A6 FE95A6   bls     FE95EE              ; Zeit in Liste kleiner ->
95AA          movea.l 26(a3),aØ           ; aØ -> Timer Stop
95AE          jsr     (aØ)                ; ---> Routine rufen
95BØ          move.l  2Ø(a4),dØ           ; dØ := tr_Secs aus Liste
95B4          cmp.l   2Ø(a5),dØ           ; = tr_Secs aus IORequest?
95B8          bne.s   FE95C2              ; nein: ->
95BA          move.l  24(a4),dØ           ; dØ := tr_Micro aus Liste
```

```
95BE          cmp.l   24(a5),d0          ; = tr_Micro aus IORequest?
95C2 FE95C2   bls     FE95E8             ; Zeit in Liste kleiner: ->
95C6          lea     20(a4),a0          ; a0 -> tr_Secs in Liste
95CA          lea     20(a5),a1          ; a1 -> tr_Secs in IORequest
95CE          bsr     FE987A             ; ---> SubTime (von Zeit in Liste)
95D2          move.l  (a3),d0            ; Macro AddHead: IORequest in Liste
95D4          move.l  a5,(a3)            ; :
95D6          movem.l d0/a3,(a5)         ; :
95DA          movea.l d0,a0              ; :
95DC          move.l  a5,4(a0)           ; :
95E0          movea.l 22(a3),a0          ; a0 -> Timerroutine
95E4          jsr     (a0)               ; ---> Timer setzen und starten
95E6          bra.s   FE963A             ; ---> Abschluß
95E8
95E8 FE95E8   movea.l 22(a3),a0          ; a0 -> Timerroutine
95EC          jsr     (a0)               ; ---> Timer setzen und starten
95EE FE95EE   lea     20(a4),a1          ; a1 -> tr_Secs in Liste
95F2          lea     20(a5),a0          ; a0 -> tr_Secs in IORequest
95F6          bsr     FE987A             ; ---> SubTime (von Zeit in IORequest)
95FA FE95FA   movea.l a4,a2              ; a2 := a4 -> ln_Succ im Request
95FC          movea.l (a2),a4            ; a4 -> nächster Request in Liste
95FE          tst.l   (a4)               ; Ende der Liste?
9600          beq.s   FE962A             ; ja: ->
9602          lea     20(a4),a1          ; a1 -> tr_Secs in Liste
9606          lea     20(a5),a0          ; a0 -> tr_Secs in IORequest
960A          move.l  (a1),d0            ; d0 := tr_Secs in Liste
960C          cmp.l   (a0),d0            ; = tr_Secs in IORequest?
960E          bne.s   FE9618             ; nein: ->
9610          move.l  4(a1),d0           ; d0 := tr_Micro in Liste
9614          cmp.l   4(a0),d0           ; = tr_Micro in IORequest?
9618 FE9618   bls     FE9624             ; Zeit in Liste kleiner: ->
961C          exg     a0,a1              ; Zeitzeiger vertauschen
961E          bsr     FE987A             ; ---> SubTime (von Zeit in Liste)
9622          bra.s   FE962A             ; ---> Request in Liste, Abschluß
9624
9624 FE9624   bsr     FE987A             ; ---> SubTime (von Zeit in IORequest)
9628          bra.s   FE95FA             ; ---> Loop
962A
962A FE962A   movea.l a3,a0              ; a0 := a3 -> Unit
962C          movea.l a5,a1              ; a1 := a5 -> IORequest
```

```
962E          move.l  a6,-(a7)            ; a6 retten
9630          movea.l D2(a6),a6           ; a6 := SysBase
9634          jsr     -EA(a6)             ; ---> Insert IORequest in Liste
9638          movea.l (a7)+,a6            ; a6 wiederherstellen
963A FE963A   movem.l (a7)+,a2/a4-a5      ; Register wiederherstellen
963E FE963E   movea.l 4,a0                ; a0 := SysBase
9642          subq.b  #1,126(a0)          ; IDNestCnt erniedrigen
9646          bge.s   FE9650              ; Interrupts noch gesperrt: ->
9648          move.w  #C000,DFF09A        ; Interrupts freigeben
9650 FE9650   movea.l (a7)+,a3            ; a3 wiederherstellen
9652          rts
9654
9654 ;--------------------------------   AbortIO
9654
9654 FE9654   movem.l a2-a3,-(a7)         ; Register retten
9658          movea.l 4,a0                ; a0 := SysBase
965C          move.w  #4000,DFF09A        ; Interrupts sperren
9664          addq.b  #1,126(a0)          ; IDNestCnt erhöhen
9668          movea.l a1,a2               ; a2 := a1 -> IORequest
966A          movea.l 18(a2),a3           ; a3 := io_Unit
966E          movea.l (a3),a0             ; a0 := lh_Head in Request Liste
9670 FE9670   move.l  (a0),d0             ; d0 := ln_Succ im nächsten Knoten
9672          beq.s   FE96A2              ; Ende der Liste: ->
9674          cmpa.l  a0,a2               ; IORequest gefunden?
9676          beq.s   FE967C              ; ja: ->
9678          movea.l d0,a0               ; a0 -> nächster Knoten
967A          bra.s   FE9670              ; ---> Loop
967C
967C FE967C   cmpa.l  (a3),a2             ; IORequest am Anfang der Liste?
967E          bne.s   FE9690              ; nein: ->
9680          movea.l 26(a3),a0           ; a0 -> Timer Stop
9684          jsr     (a0)                ; ---> Timer Stop
9686          bsr.s   FE96BC              ; ---> Request aus Liste entfernen
9688          movea.l 22(a3),a0           ; a0 -> Timer setzen und starten
968C          jsr     (a0)                ; ---> Timer setzen und starten
968E          bra.s   FE9692              ; --->
9690
9690 FE9690   bsr.s   FE96BC              ; ---> Request aus Liste entfernen
9692 FE9692   movea.l a2,a1               ; a1 := a2 -> IORequest
9694          move.b  #-2,1F(a1)          ; io_Error := ioerr_Aborted
```

```
969A          bsr     FE942A              ; ---> IORequest beantworten
969E          moveq   #0,d0               ; d0 := 0 = ok-Flag
96A0          bra.s   FE96A4              ; ---> Abschluß
96A2
96A2 FE96A2   moveq   #-1,d0              ; d0 := -1
96A4 FE96A4   movea.l 4,a0                ; a0 := SysBase
96A8          subq.b  #1,126(a0)          ; IDNestCnt erniedrigen
96AC          bge.s   FE96B6              ; Interrupts noch gesperrt: ->
96AE          move.w  #C000,DFF09A        ; Interrupts freigeben
96B6 FE96B6   movem.l (a7)+,a2-a3         ; Register wiederherstellen
96BA          rts
96BC
96BC ;------ Request aus Liste entfernen
96BC
96BC FE96BC   movea.l (a2),a0             ; Macro Remove
96BE          movea.l 4(a2),a1            ; :
96C2          move.l  a0,(a1)             ; :
96C4          move.l  a1,4(a0)            ; :
96C8          tst.l   (a0)                ; Ende der Liste?
96CA          beq.s   FE96D8              ; ja: Fertig ->
96CC          lea     20(a0),a0           ; a0 -> tr_Secs in Liste
96D0          lea     20(a2),a1           ; a1 -> tr_Secs in IORequest
96D4          bsr     FE9896              ; ---> AddTime (zur Zeit in Liste)
96D8 FE96D8   rts
96DA
96DA ;--------------------------------   InterruptServer Unit_VBlank
96DA
96DA FE96DA   movem.l a3/a6,-(a7)         ; Register retten
96DE          movea.l a1,a3               ; a3 := a1 -> Unit
96E0          movea.l 1E(a3),a6           ; a6 -> timer.device
96E4          movea.l 4,a0                ; a0 := SysBase
96E8          move.w  #4000,DFF09A        ; Interrupts sperren
96F0          addq.b  #1,126(a0)          ; IDNestCnt erhöhen
96F4          subq.w  #1,122(a0)          ; Elapsed (Task-Zeit) erniedrigen
96F8          bne.s   FE9706              ; noch nicht 0: ->
96FA          move.w  120(a0),122(a0)     ; Elapsed := Quantum = 16 Ticks
9700          ori.w   #4000,124(a0)       ; Scheduling Attention Flag setzen
9706 FE9706   lea     BFE001,a1           ; a1 -> ciaa
970C          move.b  A00(a1),d0          ; d0 := Event MSB
9710          move.b  800(a1),d1          ; d1 := Event LSB: Latch lösen
```

```
9714          tst.b   dØ                ; Event MSB = Ø?
9716          beq.s   FE974Ø            ; ja: Kein Überlauf in Sicht ->
9718          lea     C6(a6),aØ         ; aØ -> Zeitbasis im timer.device
971C          bsr     FE9494            ; ---> GetSysTime
972Ø          bsr     FE9536            ; ---> Tageszeit-Uhr zurücksetzen
9724          bra.s   FE974Ø            ; ---> Zeitzähler bedienen
9726
9726 ;--------------------------------  InterruptServer Unit_MicroHz
9726
9726 FE9726   movea.l 4,aØ              ; aØ := SysBase
972A          move.w  #4ØØØ,DFFØ9A      ; Interrupts sperren
9732          addq.b  #1,126(aØ)        ; IDNestCnt erhöhen
9736          movem.l a3/a6,-(a7)       ; Register retten
973A          movea.l a1,a3             ; a3 := a1 -> Unit
973C          movea.l 1E(a3),a6         ; a6 -> timer.device
974Ø FE974Ø   moveq   #Ø,dØ             ; dØ := Ø
9742          move.l  dØ,4Ø(a3)         ; lh_Head von Reply Liste := Ø
9746          cmpa.l  8(a3),a3          ; Request-Liste leer?
974A          bne.s   FE977Ø            ; nein: ->
974C          movea.l 26(a3),aØ         ; aØ -> Timer Stop
975Ø          jsr     (aØ)              ; ---> Routine rufen
9752 FE9752   movea.l 4,aØ              ; aØ := SysBase
9756          subq.b  #1,126(aØ)        ; IDNestCount erniedrigen
975A          bge.s   FE9764            ; Interrupts noch gesperrt: ->
975C          move.w  #C ØØØ,DFFØ9A     ; Interrupts freigeben
9764 FE9764   bsr     FE978A            ; ---> TimerRequest beantworten
9768          movem.l (a7)+,a3/a6       ; Register wiederherstellen
976C          moveq   #Ø,dØ             ; dØ := Ø
976E          rts
977Ø
977Ø FE977Ø   lea     E(a3),aØ          ; aØ -> Subtrahend für Zeitzähler
9774          bsr     FE97A2            ; ---> Zeitzähler dekrementieren
9778          movea.l 22(a3),aØ         ; aØ -> Timer setzen und starten
977C          jsr     (aØ)              ; ---> Routine rufen
977E          bra.s   FE9752            ; ---> Abschluß
978Ø
978Ø ;--------------------------------  Request in Reply-Liste stellen
978Ø
978Ø FE978Ø   move.l  4Ø(a3),(a1)       ; Request.ln_Succ := ReplyList.lh_Head
9784          move.l  a1,4Ø(a3)         ; ReplyList.lh_Head -> Request
```

```
9788          rts
978A
978A ;--------------------------------   TimerRequest beantworten
978A
978A FE978A   movea.l 40(a3),a3          ; a3 := lh_Head von ReplyList
978E          movea.l D2(a6),a6          ; a6 := SysBase
9792 FE9792   move.l  a3,d0              ; d0 := a3
9794          beq.s   FE97A0             ; = 0: Fertig ->
9796          movea.l (a3),a3            ; a3 := ln_Succ im nächsten Knoten
9798          movea.l d0,a1              ; a1 -> Request
979A          jsr     -17A(a6)           ; ---> ReplyMsg
979E          bra.s   FE9792             ; ---> Loop
97A0
97A0 FE97A0   rts
97A2
97A2 ;----------------------- Zeitzähler im ersten Request dekrementieren
97A2
97A2 FE97A2   move.l  4(a0),-(a7)        ; Subtrahend Micro auf Stack
97A6          move.l  (a0),-(a7)         ; Subtrahend Secs auf Stack
97A8 FE97A8   movea.l (a3),a0            ; a0 := lh_Head in Request-Liste
97AA          tst.l   (a0)               ; Ende der Liste?
97AC          beq.s   FE97DE             ; ja: ->
97AE          lea     20(a0),a0          ; a0 -> tr_Secs im IORequest
97B2          movea.l a7,a1              ; a1 -> Zeitdaten im Stack
97B4          move.l  (a1),d0            ; d0 := Subtrahend Secs
97B6          cmp.l   (a0),d0            ; = tr_Secs im IORequest?
97B8          bne.s   FE97C2             ; nein: ->
97BA          move.l  4(a1),d0           ; d0 := Subtrahend Micro
97BE          cmp.l   4(a0),d0           ; = tr_Micro im IORequest?
97C2 FE97C2   bcc.s   FE97CA             ; Subtrahend größer: ->
97C4          bsr     FE987A             ; ---> SubTime (von IORequest-Zeit)
97C8          bra.s   FE97DE             ; ---> Abschluß
97CA
97CA FE97CA   exg     a0,a1              ; Differenzglieder vertauschen
97CC          bsr     FE987A             ; ---> SubTime (von Subtrahend)
97D0          movea.l (a3),a1            ; Request aus Liste entfernen
97D2          movea.l (a1),a0            ; :
97D4          move.l  a0,(a3)            ; :
97D6          move.l  a3,4(a0)           ; :
97DA          bsr.s   FE9780             ; ---> Request in Reply-Liste stellen
```

```
97DC          bra.s   FE97A8              ; ---> Loop
97DE
97DE FE97DE   addq.l  #8,a7               ; Stackzeiger korrigieren
97E0          rts
97E2
97E2 ;--------------------------------   Timer-Routinen
97E2
97E2 FE97E2   bclr.b  #0,BFEF01           ; Timer B Stop
97EA          rts
97EC
97EC ;------ Timer B setzen und starten
97EC
97EC FE97EC   movea.l (a3),a1             ; a1 := lh_Head aus Request-Liste
97EE          tst.l   (a1)                ; Ende der Liste?
97F0          beq.s   FE97E2              ; ja: Timer B Stop ->
97F2          move.l  #16595,d0           ; d0 := 91541 = Timerlaufzeit in us
97F8          tst.l   20(a1)              ; tr_Secs = 0?
97FC          bne.s   FE9808              ; nein: ->
97FE          move.l  24(a1),d1           ; d1 := tr_Micro
9802          cmp.l   d0,d1               ; tr_Micro < Timerlaufzeit?
9804          bcs.s   FE980A              ; ja: ->
9806          nop
9808 FE9808   move.l  d0,d1               ; d1 := Timerlaufzeit
980A FE980A   move.l  d1,12(a3)           ; = min(tr_Micro, Timerlaufzeit)
980E          move.w  d1,d0               ; d0 := d1
9810          mulu    #B746,d0            ; d0 := d0 * 46918
9814          swap    d0                  ; d0.w := d0\65536 = nächster Timerlauf
9816          swap    d1                  ; d1.w := d1\65536 = 0 oder 1
9818          tst.w   d1                  ; d1.w = 0?
981A          beq.s   FE9820              ; ja: ->
981C          addi.w  #B746,d0            ; d0 := d0 + 46918
9820 FE9820   move.w  d0,18(a3)           ; Subtrahend := d0
9824          move.b  d0,BFE601           ; TBLO := d0.b
982A          lsr.l   #8,d0               ; d0 um 8 Bits rechtsverschieben
982C          move.b  d0,BFE701           ; TBHI := d0.b, Timer B Start
9832          rts
9834
9834 ;------ Timerroutine VBlank
9834 FE9834   rts
```

```
9836
9836 ;------ Timer B Stop
9836
9836 FE9836  lea     BFE001,a1           ; a1 -> ciaa.pra
983C         bclr.b  #0,F00(a1)          ; Timer B Stop
9842         beq.s   FE9878              ; Timer B war schon gestoppt: ->
9844         movea.l (a3),a0             ; a0 := lh_Head aus Request-Liste
9846         tst.l   (a0)                ; Ende der Liste?
9848         bra     FE9878              ; ---> rts
984C
984C ;------ Entwicklermüll, sollte entfernt werden
984C
984C         beq.s   FE9878
984E         move.b  700(a1),d0          ; d0.b := TBHI
9852         lsl.w   #8,d0               ; um 8 Bits nach links schieben
9854         move.b  600(a1),d0          ; d0.b := TBLO
9858         sub.w   18(a3),d0           ; - Subtrahend
985C         neg.w   d0                  ; Ergebnis negieren
985E         beq.s   FE9878              ; = 0: ->
9860         mulu    #B2CB,d0            ; * 45771
9864         lsr.l   #8,d0               ; \ 32768 = Zeitrest in µs
9866         lsr.l   #7,d0
9868         move.l  d0,-(a7)            ; Ergebnis als tv_Micro auf Stack
986A         clr.l   -(a7)               ; tv_Secs = 0 auf Stack
986C         movea.l a7,a1               ; a1 -> Zeitdaten im Stack
986E         lea     20(a0),a0           ; a0 -> Zeitdaten in Request
9872         bsr     FE987A              ; ---> SubTime
9876         addq.l  #8,a7               ; Stackzeiger korrigieren
9878
9878 ;------ VBlank Timer Stop
9878
9878 FE9878  rts
987A
987A ;--------------------------------   SubTime
987A
987A FE987A  movem.l (a0),d0-d1          ; Zielzeit nach d0, d1
987E         sub.l   (a1),d0             ; Sekunden subtrahieren
9880         sub.l   4(a1),d1            ; Mikrosekunden subtrahieren
9884 FE9884  bge.s   FE9890              ; Ergebnis nicht negativ: ->
9886         subq.l  #1,d0               ; Sekunden um 1 vermindern
```

```
9888          addi.l  #0F4240,d1          ; Mikrosekunden um 1000000 vergrößern
988E          bra.s   FE9884              ; ---> Loop
9890
9890 FE9890   movem.l d0-d1,(a0)          ; Ergebnis in Zielzeit schreiben
9894          rts
9896
9896 ;--------------------------------   AddTime
9896
9896 FE9896   move.l  d2,-(a7)            ; d2 retten
9898          move.l  #0F4240,d2          ; d2 := 1000000
989E          movem.l (a0),d0-d1          ; Zielwerte nach d0, d1
98A2          add.l   (a1),d0             ; Sekunden addieren
98A4          add.l   4(a1),d1            ; Mikrosekunden addieren
98A8 FE98A8   cmp.l   d1,d2               ; Mikrosekunden < 1000000?
98AA          bgt.s   FE98B2              ; ja: ->
98AC          addq.l  #1,d0               ; Sekunden um 1 erhöhen
98AE          sub.l   d2,d1               ; Mikrosekunden um 1000000 vermindern
98B0          bra.s   FE98A8              ; ---> Loop
98B2
98B2 FE98B2   movem.l d0-d1,(a0)          ; Ergebniswerte zurückspeichern
98B6          move.l  (a7)+,d2            ; d2 wiederherstellen
98B8          rts
98BA
98BA ;--------------------------------   CmpTime
98BA
98BA FE98BA   moveq   #0,d0               ; d0 := 0
98BC          move.l  (a1),d1             ; d1 := Secs1
98BE          cmp.l   (a0),d1             ; vergleichen mit Secs0
98C0          bne.s   FE98CA              ; ungleich: ->
98C2          move.l  4(a1),d1            ; d1 := Micro1
98C6          cmp.l   4(a0),d1            ; vergleichen mit Micro0
98CA FE98CA   beq.s   FE98D0              ; gleich: Ausgang mit d0 = 0 ->
98CC          bhi.s   FE98D2              ; Zeit1 > Zeit0: ->
98CE          moveq   #-1,d0              ; d0 := -1
98D0 FE98D0   rts
98D2
```

```
98D2 FE98D2  moveq   #1,d0                ; d0 := 1
98D4         bra.s   FE98D0               ; ---> rts
98D6
98D6         DC.W    0
98D8
98D8 ************************************************************************
```

Trackdisk.device Datenbereich

```
-Ø24 ************************* Trackdisk.device *************************
-Ø24
-Ø24          jmp     FEAØ6A      ; ---> AbortIO
-Ø1E          jmp     FE9FBE      ; ---> BeginIO
-Ø18          jmp     FEAØ6A      ; ---> Extfunct
-Ø12          jmp     FE9F14      ; ---> Expunge
-ØØC          jmp     FE9F92      ; ---> Close
-Ø06          jmp     FE9F42      ; ---> Open
ØØØØ
ØØØØ ;-------------------------   Library-Block
ØØØØ
ØØØØ          DC.L    Ø7D8        ; ln_Succ
ØØØ4          DC.L    45B8        ; ln_Pred
ØØØ8          DC.B    3           ; ln_Type
ØØØ9          DC.B    Ø           ; ln_Pri
ØØØA          DC.L    FE98FE      ; ln_Name
ØØØE
ØØØE          DC.B    4           ; lib_Flags
ØØØF          DC.B    Ø           ; lib_Pad
ØØ1Ø          DC.W    24          ; lib_NegSize
ØØ12          DC.W    A8          ; lib_PosSize
ØØ14          DC.W    21          ; lib_Version
ØØ16          DC.W    7F          ; lib_Revision
ØØ18          DC.L    Ø           ; lib_IdString
ØØ1C          DC.L    9E44ØØØØ    ; lib_Sum
ØØ2Ø          DC.W    2           ; lib_OpenCnt
ØØ22
ØØ22          DC.B    Ø           ; Flags (Bit 4: Reset, Bit 5: Write)
ØØ23          DC.B    Ø
ØØ24
```

```
0024 ;-------------------------   UnitPointer
0024
0024          DC.L    6038        ; Unit 0
0028          DC.L    CE78        ; Unit 1
002C          DC.L    0           ; Unit 2
0030          DC.L    0           ; Unit 3
0034
0034 ;-------------------------   Diverse Adressen
0034
0034          DC.L    0676        ; SysBase
0038          DC.L    21FE        ; graphics.library
003C          DC.L    1EDE        ; disk.resource
0040          DC.L    0
0044          DC.L    0
0048          DC.L    0
004C          DC.W    0
004E          DC.L    2EDE        ; timer.device
0052          DC.L    2F08        ; unit_MicroHz
0056          DC.L    2EDE        ; timer.device
005A          DC.L    2F56        ; unit_VBlank
005E          DC.L    1B18        ; ciab.resource
0062
0062 ;-------------------------   ResetHandler
0062
0062     00   DC.L    2D14        ; ln_Succ
0066     04   DC.L    2D10        ; ln_Pred
006A     08   DC.B    0           ; ln_Type
006B     09   DC.B    0           ; ln_Pri
006C     0A   DC.L    0           ; ln_Name
0070     0E   DC.L    *0000       ; is_Data
0074     12   DC.L    FE9EEE      ; is_Code
0078
0078 ;-------------------------   IORequest (Keyboard, für ResetHandlerDone)
0078
0078     00   DC.L    0           ; ln_Succ
007C     04   DC.L    0           ; ln_Pred
0080     08   DC.B    0           ; ln_Type
0081     08   DC.B    0           ; ln_Pri
0082     0C   DC.L    0           ; ln_Name
0086     10   DC.L    0           ; mn_ReplyPort
```

```
008A     10  DC.W    0           ; mn_Length
008C     14  DC.L    2C1C        ; io_Device -> keyboard.device
0090     18  DC.L    2C50        ; io_Unit -> MsgPort im keyboard.device
0094     1C  DC.W    D           ; io_Command = 'kbd_ResetHandlerDone'
0096     1E  DC.B    1           ; io_Flags
0097     1F  DC.B    0           ; io_Error
0098     20  DC.L    0           ; io_Actual
009C     24  DC.L    0           ; io_Length
00A0     28  DC.L    46F6        ; io_Data
00A4     2C  DC.L    0           ; io_Offset
00A8
00A8 ************************************************************************
```

Trackdisk.unit Datenbereich

```
0000 **************  trackdisk.unit  ************************************
0000
0000 ;-------------------------   MessagePort
0000
0000          DC.L                ; ln_Succ
0004          DC.L                ; ln_Pred
0008          DC.B    4           ; ln_Type (4 = nt_MsgPort)
0009          DC.B                ; ln_Pri
000A          DC.L    FE98FE      ; ln_Name
000E
000E          DC.B    0           ; mp_Flags
000F          DC.B    9           ; mp_SigBit
0010          DC.L    *012E       ; mp_SigTask
0014          DC.L    *0018       ; lh_Head        \  mp_MsgList
0018          DC.L    0           ; lh_Tail        |
001C          DC.L    *0014       ; lh_TailPred    |
0020          DC.B    5           ; lh_Type        |
0021          DC.B    0           ; lh_Pad         /
0022
0022 ;-------------------------   Verschiedene Parameter
0022
0022          DC.B    0           ; unit_Flags
0023          DC.B    0           ; unit_Pad
0024          DC.W    1           ; unit_OpenCnt
0026
0026          DC.W    50          ; = 80   = tdu_Comp01Track
0028          DC.W    FFFF        ; = 65535= tdu_Comp10Track
002A          DC.W    FFFF        ; = 65535= tdu_Comp11Track
002C          DC.L    BB8         ; = 3000 = tdu_StepDelay
0030          DC.L    3A98        ; = 6000 = tdu_SettleDelay
0034          DC.B    A           ; = 10   = tdu_RetryCnt
0035          DC.B    1           ; = Typ Drive3_5
```

```
0036          DC.W    A0          ; = 160    (Spuren pro Disk)
0038          DC.L    DC000       ; = 901120 (Bytes pro Disk)
003C          DC.L    0           ; Laufwerk-Typ
0040
0040 ;-------------------------   Lese- und Schreibparameter
0040
0040          DC.B    2           ; Flags:    Bit 0  SecLabel
0041                              ;                1  NoDisk
0041                              ;                2  ExtCmd
0041                              ;                3  Closed
0041                              ;                4  WrProt
0041
0041          DC.B    F3          ; Disk-Kontrollbyte:
0042                              ;           Bit 0  STEP (Puls lo-hi)
0042                              ;                1  DIR  (0: nach innen)
0042                              ;                2  SIDE (0: oben)
0042                              ;                3  SEL0 (0: selektiert)
0042                              ;                4  SEL1
0042                              ;                5  SEL2
0042                              ;                6  SEL3
0042                              ;                7  MTR  (0: ein)
0042          DC.B    0           ; Sektor-Zähler
0043          DC.B    0           ; UnitNum
0044          DC.L    108C        ; -> IORequest
0048          DC.B    0           ;
0049          DC.B    4           ; Sektornummer (aus IORequest)
004A          DC.W    6D          ; Spurnummer (aus IORequest)
004C          DC.W    -1          ; Nummer der Spur unter Kopf
004E
004E          DC.L    63D8        ; -> Spurpuffer
0052          DC.L    63D8        ; -> Spurpuffer
0056          DC.L    B254        ; io_Data
005A          DC.L    0           ; iotd_SecLabel
005E
005E ;-------------------------   TimeRequest unit_MicroHz
005E
005E     00   DC.L    *00C6       ; ln_Succ
0062     04   DC.L    *00C2       ; ln_Pred
0066     08   DC.B    5           ; ln_Type
0067     09   DC.B    0           ; ln_Pri
```

```
0068     0A  DC.L    FE98FE      ; ln_Name
006C     0E  DC.L    *00AE       ; mn_ReplyPort
0070     12  DC.W    0           ; mn_Length
0072     14  DC.L    2EDE        ; io_Device -> timer.device
0076     18  DC.L    2F08        ; io_Unit -> unit_MicroHz
007A     1C  DC.W    9           ; io_Command = tr_AddRequest
007C     1E  DC.B    0           ; io_Flags
007D     1F  DC.B    0           ; io_Error
007E     20  DC.L    0           ; tr_Secs
0082     24  DC.L    3E8         ; tr_Micro
0086
0086 ;------------------------   TimeRequest unit_VBlank
0086
0086     00  DC.L    4374        ; ln_Succ
008A     04  DC.L    2F56        ; ln_Pred
008E     08  DC.B    5           ; ln_Type
008F     09  DC.B    0           ; ln_Pri
0090     0A  DC.L    FE98FE      ; ln_Name
0094     0E  DC.L    *0000       ; mn_ReplyPort
0098     12  DC.W    0           ; mn_Length
009A     14  DC.L    2EDE        ; io_Device -> timer.device
009E     18  DC.L    2F56        ; io_Unit -> unit_VBlank
00A2     1C  DC.W    9           ; io_Command
00A4     1E  DC.B    0           ; io_Flags
00A5     1F  DC.B    0           ; io_Error
00A6     20  DC.L    0           ; tr_Secs
00AA     24  DC.L    9C40        ; tr_Micro
00AE
00AE ;------------------------   ReplyPort für Laufwerk-Anforderung
00AE
00AE     00  DC.L    0           ; ln_Succ
00B2     04  DC.L    0           ; ln_Pred
00B6     08  DC.B    4           ; ln_Type (4 = nt_MsgPort)
00B7     09  DC.B    0           ; ln_Pri
00B8     0A  DC.L    FE98FE      ; ln_Name -> 'trackdisk.device'
00BC     0E  DC.B    0           ; mp_Flags
00BD     0F  DC.B    A           ; mp_SigBit
00BE     10  DC.L    *012E       ; mp_SigTask
00C2     14  DC.L    *00C6       ; lh_Head
00C6     18  DC.L    0           ; lh_Tail
```

```
00CA     1C  DC.L    *00C2       ; lh_TailPred
00CE     20  DC.B    5           ; lh_Type
00CF     21  DC.B    0           ; lh_Pad
00D0
00D0 ;-------------------------   DiskResource-Unit
00D0
00D0 ;------ Message für Laufwerk-Anforderung
00D0
00D0     00  DC.L    *00C6       ; ln_Succ
00D4     04  DC.L    *00C2       ; ln_Pred
00D8     08  DC.B    7           ; ln_Type
00D9     09  DC.B    0           ; ln_Pri
00DA     0A  DC.L    FE98FE      ; ln_Name
00DE     0E  DC.L    *00AE       ; mn_ReplyPort
00E2     12  DC.W    0           ; mn_Length
00E4
00E4 ;------ Interrupt-Struktur für DskBlk-Interrupt
00E4
00E4     14  DC.L    0           ; ln_Succ
00E8     18  DC.L    0           ; ln_Pred
00EC     1C  DC.B    0           ; ln_Type
00ED     1D  DC.B    0           ; ln_Pri
00EE     1E  DC.L    0           ; ln_Name
00F2     22  DC.L    *0000       ; is_Data
00F6     26  DC.L    FEA6F2      ; is_Code
00FA
00FA ;------ Interrupt-Struktur für DskSync-Interrupt
00FA
00FA     2A  DC.L    0           ; ln_Succ
00FE     2E  DC.L    0           ; ln_Pred
0102     32  DC.B    0           ; ln_Type
0103     33  DC.B    0           ; ln_Pri
0104     34  DC.L    0           ; ln_Name
0108     38  DC.L    0           ; is_Data
010C     3C  DC.L    0           ; is_Code
0110
0110 ;------ Interrupt-Struktur für Index-Interrupt
0110
0110     40  DC.L    0           ; ln_Succ
0114     44  DC.L    0           ; ln_Pred
```

```
0118     48  DC.B    0           ; ln_Type
0119     49  DC.B    0           ; ln_Pri
011A     4A  DC.L    0           ; ln_Name
011E     4E  DC.L    0           ; is_Data
0122     52  DC.L    FEB38E      ; is_Code
0126
0126 ;-------------------------   Diskwechsel-Parameter
0126
0126         DC.L    1           ; Diskwechsel-Zähler
012A         DC.L    106C        ; -> DiskChange-Interrupt-Struktur
012E
012E ;-------------------------   Trackdisk-Task-Kontrollblock
012E
012E     00  DC.L    32D2        ; ln_Succ
0132     04  DC.L    CFA6        ; ln_Pred
0136     08  DC.B    1           ; ln_Type (1 = nt_Task)
0137     09  DC.B    5           ; ln_Pri
0138     0A  DC.L    FE98FE      ; ln_Name
013C     0E  DC.B    0           ; tc_Flags
013D     0F  DC.B    4           ; tc_State
013E     10  DC.B    0           ; tc_IDNestCnt
013F     11  DC.B    -1          ; tc_TDNestCnt
0140     12  DC.L    FFFF        ; tc_SigAlloc
0144     16  DC.L    0300        ; tc_SigWait
0148     1A  DC.L    0           ; tc_SigRecvd
014C     1E  DC.L    0           ; tc_SigExcept
0150     22  DC.W    8000        ; tc_TrapAlloc
0152     24  DC.W    0           ; tc_TrapAble
0154     26  DC.L    0           ; tc_ExceptData
0158     2A  DC.L    FC2FB4      ; tc_ExceptCode
015C     2E  DC.L    0           ; tc_TrapData
0160     32  DC.L    FC2FB4      ; tc_TrapCode
0164     36  DC.L    637E        ; tc_SPReg
0168     3A  DC.L    61D8        ; tc_SPLower
016C     3E  DC.L    63D8        ; tc_SPUpper
0170     42  DC.L    0           ; tc_Switch
0174     46  DC.L    0           ; tc_Launch
0178     4A  DC.L    6010        ; lh_Head       \  MemList
017C     4E  DC.L    0           ; lh_Tail       |
0180     52  DC.L    6010        ; lh_TailPred   |
```

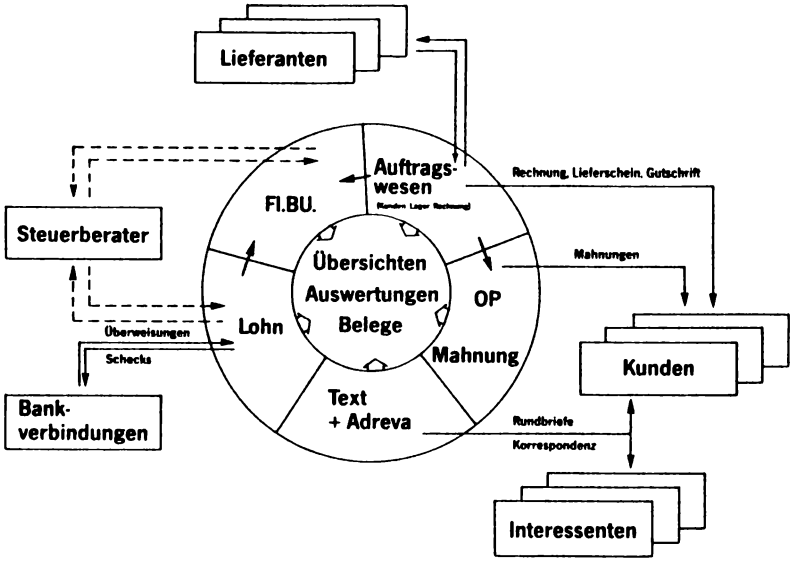
```
0184     56  DC.B    0           ; lh_Type        |
0185     57  DC.B    0           ; lh_Pad         /
0186     58  DC.L    0           ; tc_UserData
018A
018A ;------
018A
018A         DC.L    6010        ; -> ME-Struktur
018A
018E ;-------------------------   ChangeInt-Liste
018E
018E     00  DC.L    *0192       ; lh_Head
0192     04  DC.L    0           ; lh_Tail
0196     08  DC.L    *018E       ; lh_TailPred
019A     0C  DC.B                ; lh_Type
019B     0D  DC.B                ; lh_Pad
019C
019C ************************************************************************
```

COMMODORE
AMIGA

GIV

Wir bieten das kaufmännische Programm für die AMIGA - Familie sowie alle IBM PC/AT kompatiblen Rechner - ein professionelles Paket, das durch seine **Bedienerfreundlichkeit**, **Schnelligkeit** und **Sicherheit** besticht.

Auftragswesen	798.-	DM	**Finanzbuchhaltung**	798.-	DM
Offene Posten/Mahnung	180.-	DM	**Lohnbuchhaltung**	798.-	DM
Text/Adreßverwaltung	342.-	DM	**Integration Auftr./Fibu**	550.-	DM



Achtung Wiederverkäufer: Sie können das in C geschriebene modulare Programmpaket unter eigenem Label zu einer Einmal-Lizenz erwerben.

GIV, Scherbaumstraße 29, D-8000 München 83, Telefon 089/6375726

Trackdisk.device Routinen

```
98E4 *************** Trackdisk.device-Routinen ****************************
98E4
98E4 FE98E4  DC.W    4AFC                ; rt_Matchword
98E6         DC.L    FE98E4              ; rt_Matchtag
98EA         DC.L    FEB3DC              ; rt_EndSkip
98EE         DC.B    1                   ; rt_Flags
98EF         DC.B    21                  ; rt_Version
98F0         DC.B    3                   ; rt_Type
98F1         DC.B    14                  ; rt_Pri
98F2         DC.L    FE98FE              ; rt_Name
98F6         DC.L    FE9910              ; rt_IdString
98FA         DC.L    FE9B3E              ; rt_Init
98FE
98FE FE98FE  DC.B    'trackdisk.device',0,0
9910
9910 FE9910  DC.B    'trackdisk 33.127 (8 Oct 1986)',0D,0A,0
9930
9930 ;--------------------------------   Device öffnen (nicht gerufen)
9930
9930 FE9930  lea     -30(a7),a7          ; 48 Bytes im Stack freimachen
9934         movea.l a7,a1               ; a1 := a7 -> IOStdRequest im Stack
9936         lea     -3A(pc),a0          ; a0 := FE98FE -> 'trackdisk.device'
993A         move.l  a0,A(a1)            ; a0 in ln_Name eintragen
993E         moveq   #0,d0               ; d0 := 0 = UnitNum
9940         moveq   #0,d1               ; d1 := 0 = Flags
9942         jsr     -1BC(a6)            ; ---> OpenDevice
9946         tst.l   d0                  ; Device geöffnet?
9948         beq.s   FE995A              ; ja: Fertig ->
994A         movea.l a7,a1               ; a1 := a7 -> IOStdRequest im Stack
994C         move.l  14(a1),-(a7)        ; io_Device auf Stack legen
9950         jsr     -1C2(a6)            ; ---> CloseDevice
9954         movea.l (a7)+,a1            ; a1 := io_Device
```

```
9956          jsr     -1B6(a6)            ; ---> RemDevice
995A FE995A   lea     30(a7),a7           ; Platz im Stack wieder freigeben
995E          rts
9960
9960 ;--------------------------------   Laufwerkprüfung, Zeitsignal
9960
9960 FE9960   bsr     FE998C              ; ---> Prüfen ob Disk im Laufwerk
9964          lea     86(a3),a1           ; a1 -> IORequest unit_VBlank
9968          move.l  #07A120,24(a1)      ; tr_Micro = 500000
9970          btst.b  #1,40(a3)           ; NoDisk-Flag gesetzt?
9976          beq.s   FE997E              ; nein: ->
9978          moveq   #2,d0               ; d0 := 2
997A          move.l  d0,20(a1)           ; tr_Secs := d0
997E FE997E   move.l  a6,-(a7)            ; a6 retten
9980          movea.l 34(a6),a6           ; a6 := SysBase
9984          jsr     -1CE(a6)            ; ---> SendIO
9988          movea.l (a7)+,a6            ; a6 wiederherstellen
998A          rts
998C
998C ;--------------------------------   Prüfen ob Disk im Laufwerk
998C
998C FE998C   movem.l d2/a2,-(a7)         ; Register retten
9990          bsr     FEADDC              ; ---> Laufwerk anfordern
9994          lea     BFD100,a2           ; a2 -> ciab.prb = Disk-Kontrollregister
999A          move.b  41(a3),(a2)         ; Kontrollbyte schreiben
999E          nop                         ; ca. 1 us warten
99A0          nop
99A2          move.b  BFE001,d2           ; d2 := Diskstatus
99A8          bsr     FEAE42              ; ---> Laufwerk freigeben
99AC          btst.l  #2,d2               ; DSKCHANGE gesetzt?
99B0          beq     FE99C2              ; ja: Keine Disk im Laufwerk ->
99B4          btst.b  #1,40(a3)           ; NoDisk-Flag gesetzt?
99BA          beq     FE9A62              ; nein: ->
99BE          bra     FE9A0C              ; ---> Diskwechsel bearbeiten
99C2
99C2 ;------ Keine Disk im Laufwerk
99C2
99C2 FE99C2   bset.b  #1,40(a3)           ; NoDisk-Flag setzen
99C8          bne     FE99E8              ; war schon gesetzt: ->
99CC          addq.l  #1,126(a3)          ; DiskChange-Zähler erhöhen
```

```
99DØ          moveq   #6,dØ               ; dØ := 6
99D2          bsr     FEA3DA              ; ---> Kopf auf Spur dØ setzen
99D6          moveq   #Ø,dØ               ; dØ := Ø
99D8          bsr     FEA462              ; ---> Motor ausschalten
99DC          bsr     FE9A68              ; ---> DiskChange-Interrupt erzeugen
99EØ          bsr     FEADDC              ; ---> Laufwerk anfordern
99E4          move.b  41(a3),(a2)         ; Kontrollbyte schreiben
99E8
99E8 FE99E8   bchg.b  #1,41(a3)           ; DSKDIREC-Bit flippen
99EE          move.b  41(a3),dØ           ; dØ := Kontrollbyte
99F2          move.b  dØ,d1               ; d1 := dØ
99F4          bclr.l  #Ø,d1               ; DSKSTEP-Bit := Ø
99F8          move.b  d1,(a2)             ; Kontrollbyte schreiben
99FA          nop                         ; ca. 1 us warten
99FC          nop
99FE          move.b  dØ,(a2)             ; DSKSTEP-Bit := 1
9AØØ          bsr     FEAE42              ; ---> Laufwerk freigeben
9AØ4          move.l  2C(a3),dØ           ; dØ := tdu_StepDelay
9AØ8          bsr     FEA4FØ              ; ---> dØ * 2 us warten
9AØC FE9AØC   bsr     FEADDC              ; ---> Laufwerk anfordern
9A1Ø          move.b  41(a3),(a2)         ; Kontrollbyte schreiben
9A14          nop                         ; ca 1 us warten
9A16          nop
9A18          move.b  BFEØØ1,d2           ; d2 := Diskstatus
9A1E          bsr     FEAE42              ; ---> Laufwerk freigeben
9A22          btst.l  #2,d2               ; DSKCHANGE gesetzt?
9A26          beq     FE9A62              ; ja: Keine Disk im Laufwerk ->
9A2A          bsr     FEADDC              ; ---> Laufwerk anfordern
9A2E          move.b  41(a3),(a2)         ; Kontrollbyte schreiben
9A32          move.b  BFEØØ1,d2           ; d2 := Diskstatus
9A38          bsr     FEAE42              ; ---> Laufwerk freigeben
9A3C          addq.l  #1,126(a3)          ; DiskChange-Zähler erhöhen
9A4Ø          bclr.b  #4,4Ø(a3)           ; Schreibschutz-Flag löschen
9A46          btst.l  #3,d2               ; DSKPROT (Schreibschutz) gesetzt?
9A4A          bne.s   FE9A52              ; nein: ->
9A4C          bset.b  #4,4Ø(a3)           ; Schreibschutz-Flag setzen
9A52 FE9A52   move.w  #-1,4C(a3)          ; Spurnummer ungültig machen
9A58          bclr.b  #1,4Ø(a3)           ; NoDisk-Flag rücksetzen
9A5E          bsr     FE9A68              ; ---> DiskChange-Interrupt erzeugen
```

```
9A62 FE9A62  movem.l (a7)+,d2/a2          ; Register wiederherstellen
9A66         rts
9A68
9A68 ;------ Diskwechsel-Interrupt erzeugen
9A68
9A68 FE9A68  movem.l d2/a6,-(a7)          ; Register retten
9A6C         movea.l 34(a6),a6            ; a6 := SysBase
9A70         move.l  12A(a3),dØ           ; dØ -> ChangeInt-Struktur
9A74         beq.s   FE9A7C               ; Keine Adresse eingetragen: ->
9A76         movea.l dØ,a1                ; a1 := dØ -> ChangeInt-Struktur
9A78         jsr     -B4(a6)              ; ---> Cause
9A7C FE9A7C  move.l  18E(a3),d2           ; d2 := lh_Head von DiskChangeInt Liste
9A8Ø FE9A8Ø  movea.l d2,a1                ; a1 := d2 -> nächster Knoten
9A82         move.l  (a1),d2              ; d2 := ln_Succ
9A84         beq.s   FE9A9Ø               ; = Ø: Ende der Liste ->
9A86         movea.l 28(a1),a1            ; a1 := io_Data -> ChangeInt-Struktur
9A8A         jsr     -B4(a6)              ; ---> Cause
9A8E         bra.s   FE9A8Ø               ; ---> Loop
9A9Ø
9A9Ø FE9A9Ø  movem.l (a7)+,d2/a6          ; Register wiederherstellen
9A94         rts
9A96
9A96 ;--------------------------------   td_ChangeNum
9A96
9A96 FE9A96  move.l  126(a3),2Ø(a1)       ; io_Actual := DiskChange-Zähler
9A9C         bsr     FEA1BØ               ; ---> IORequest beantworten
9AAØ         rts
9AA2
9AA2 ;--------------------------------   td_ChangeState
9AA2
9AA2 FE9AA2  moveq   #Ø,dØ                ; dØ := Ø (überflüssig)
9AA4         btst.b  #1,4Ø(a3)            ; NoDisk-Flag gesetzt?
9AAA         sne     dØ                   ; ja: dØ := -1, nein: dØ := Ø
9AAC         move.l  dØ,2Ø(a1)            ; io_Actual := dØ
9ABØ         bsr     FEA1BØ               ; ---> IORequest beantworten
9AB4         rts
9AB6
9AB6 ;--------------------------------   td_Remove
9AB6
9AB6 FE9AB6  move.l  28(a1),12A(a3)       ; ChangeInt-Adresse := io_Data
```

```
9ABC          bsr     FEA1BØ              ; ---> IORequest beantworten
9ACØ          rts
9AC2
9AC2 ;--------------------------------   td_AddChangeInt
9AC2
9AC2 FE9AC2   lea     18E(a3),aØ          ; aØ -> ChangeInt List Header
9AC6          lea     4(aØ),aØ            ; Macro AddTail
9ACA          move.l  4(aØ),dØ            ; :
9ACE          move.l  a1,4(aØ)            ; :
9AD2          move.l  aØ,(a1)             ; :
9AD4          move.l  dØ,4(a1)            ; :
9AD8          movea.l dØ,aØ               ; :
9ADA          move.l  a1,(aØ)             ; :
9ADC          rts
9ADE
9ADE ;--------------------------------   td_RemChangeInt
9ADE
9ADE FE9ADE   move.l  a1,d1               ; a1 in d1 retten
9AEØ          movea.l (a1),aØ             ; Macro Remove
9AE2          movea.l 4(a1),a1            ; :
9AE6          move.l  aØ,(a1)             ; :
9AE8          move.l  a1,4(aØ)            ; :
9AEC          movea.l d1,a1               ; a1 wiederherstellen
9AEE          bsr     FEA1BØ              ; ---> IORequest beantworten
9AF2          rts
9AF4
9AF4 ;--------------------------------   Namen
9AF4
9AF4 FE9AF4   DC.B    'disk.resource',Ø
9BØ2
9BØ2 FE9BØ2   DC.B    'ciab.resource',Ø
9B1Ø
9B1Ø FE9B1Ø   DC.B    'graphics.library',Ø
9B21
9B21 FE9B21   DC.B    'timer.device',Ø
9B2E
9B2E FE9B2E   DC.B    'keyboard.device',Ø
9B3E
```

```
9B3E ;--------------------------------   Initialisierung
9B3E
9B3E FE9B3E  move.l  a2,-(a7)           ; a2 retten
9B4Ø         link    a4,#-8C            ; Stackrahmen von 14Ø Bytes schaffen
9B44         movea.l a7,a2              ; a2 -> Stackrahmen
9B46         lea     -38(pc),a1         ; a1 := FE9B1Ø -> 'graphics.library'
9B4A         moveq   #Ø,dØ              ; dØ := Ø = Version
9B4C         jsr     -228(a6)           ; ---> OpenLibrary
9B5Ø         move.l  dØ,Ø(a2)           ; GfxBase in Stackrahmen eintragen
9B54         bne.s   FE9B7Ø             ; Library ok: ->
9B56         movem.l d7/a5-a6,-(a7)     ; Register retten
9B5A         move.l  #14Ø38ØØ2,d7       ; d7 := AlertKode
9B6Ø         movea.l 4,a6               ; a6 := SysBase
9B64         jsr     -6C(a6)            ; ---> Alert
9B68         movem.l (a7)+,d7/a5-a6     ; Register wiederherstellen
9B6C         bra     FE9C58             ; ---> Fehlerausgang
9B7Ø
9B7Ø FE9B7Ø  lea     -7E(pc),a1         ; a1 := FE9AF4 -> 'disk.resource'
9B74         jsr     -1F2(a6)           ; ---> OpenResource
9B78         move.l  dØ,84(a2)          ; Disk.resource Base eintragen
9B7C         bne.s   FE9B98             ; Resource ok: ->
9B7E         movem.l d7/a5-a6,-(a7)     ; Register retten
9B82         move.l  #14Ø58Ø21,d7       ; d7 := AlertKode
9B88         movea.l 4,a6               ; a6 := SysBase
9B8C         jsr     -6C(a6)            ; ---> Alert
9B9Ø         movem.l (a7)+,d7/a5-a6     ; Register wiederherstellen
9B94         bra     FE9C5Ø             ; ---> Fehlerausgang
9B98
9B98 FE9B98  lea     -98(pc),a1         ; a1 := FE9BØ2 -> 'ciab.resource'
9B9C         jsr     -1F2(a6)           ; ---> OpenResource
9BAØ         move.l  dØ,88(a2)          ; ciab.resource Base eintragen
9BA4         bne.s   FE9BCØ             ; Resource ok: ->
9BA6         movem.l d7/a5-a6,-(a7)     ; Register retten
9BAA         move.l  #14Ø58Ø2Ø,d7       ; d7 := AlertKode
9BBØ         movea.l 4,a6               ; a6 := SysBase
9BB4         jsr     -6C(a6)            ; ---> Alert
9BB8         movem.l (a7)+,d7/a5-a6     ; Register wiederherstellen
9BBC         bra     FE9C5Ø             ; ---> Fehlerausgang
9BCØ
9BCØ FE9BCØ  lea     -94(pc),aØ         ; aØ := FE9B2E -> 'keyboard.device'
```

```
9BC4            lea     54(a2),a1           ; a1 -> IORequest
9BC8            moveq   #0,d0               ; d0 := 0 = UnitNum
9BCA            moveq   #0,d1               ; d1 := 0 = Flags
9BCC            jsr     -1BC(a6)            ; ---> OpenDevice
9BD0            tst.l   d0                  ; Keyboard.device ok?
9BD2            beq.s   FE9BEE              ; ja: ->
9BD4            movem.l d7/a5-a6,-(a7)      ; Register retten
9BD8            move.l  #14048013,d7        ; d7 := AlertKode
9BDE            movea.l 4,a6                ; a6 := SysBase
9BE2            jsr     -6C(a6)             ; ---> Alert
9BE6            movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
9BEA            bra     FE9C50              ; ---> Fehlerausgang
9BEE
9BEE FE9BEE     lea     4(a2),a1            ; a1 -> IORequest Unit_MicroHz
9BF2            moveq   #0,d0               ; d0 := 0 = UnitNum
9BF4            bsr     FE9CE8              ; ---> timer.device öffnen
9BF8            bne.s   FE9C48              ; Fehler: ->
9BFA            lea     2C(a2),a1           ; a1 -> IORequest Unit_VBlank
9BFE            moveq   #1,d0               ; d0 := 1 = UnitNum
9C00            bsr     FE9CE8              ; ---> timer.device öffnen
9C04            bne.s   FE9C40              ; Fehler: ->
9C06            lea     FEA2E4,a0           ; a0 -> Vektortabelle
9C0C            lea     FEA1F6,a1           ; a1 -> Tabelle für InitStruct
9C12            suba.l  a2,a2               ; a2 := 0: Keine Init-Routine
9C14            move.l  #A8,d0              ; d0 := 168 Bytes Speicherbedarf
9C1A            jsr     -54(a6)             ; ---> MakeLibrary
9C1E            tst.l   d0                  ; Library ok?
9C20            bne.s   FE9C5E              ; ja: ->
9C22            movem.l d7/a5-a6,-(a7)      ; Register retten
9C26            move.l  #14020000,d7        ; d7 := AlertKode
9C2C            movea.l 4,a6                ; a6 := SysBase
9C30            jsr     -6C(a6)             ; ---> Alert
9C34            movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
9C38            lea     2C(a2),a1           ; a1 -> IORequest Unit_VBlank
9C3C            jsr     -1C2(a6)            ; ---> CloseDevice
9C40 FE9C40     lea     4(a2),a1            ; a1 -> IORequest Unit_MicroHz
9C44            jsr     -1C2(a6)            ; ---> CloseDevice
9C48 FE9C48     lea     54(a2),a1           ; a1 -> IORequest Keyboard
9C4C            jsr     -1C2(a6)            ; ---> CloseDevice
9C50 FE9C50     movea.l 0(a2),a1            ; a1 := GfxBase
```

```
9C54          jsr     -19E(a6)          ; ---> CloseLibrary
9C58 FE9C58   moveq   #-1,d0            ; d0 := -1 = Fehlerflag
9C5A          bra     FE9CE2            ; ---> Abschluß
9C5E
9C5E FE9C5E   lea     -8C(a4),a0        ; a0 -> Stackrahmen
9C62          movea.l d0,a2             ; a2 := d0 -> trackdisk.device
9C64          move.l  0(a0),38(a2)      ; GfxBase eintragen
9C6A          move.l  84(a0),3C(a2)     ; disk.resource eintragen
9C70          move.l  88(a0),5E(a2)     ; ciab.resource eintragen
9C76          move.l  a6,34(a2)         ; SysBase eintragen
9C7A          move.l  -74(a4),4E(a2)    ; timer.device eintragen
9C80          move.l  -70(a4),52(a2)    ; unit_MicroHz eintragen
9C86          move.l  -4C(a4),56(a2)    ; timer.device eintragen
9C8C          move.l  -48(a4),5A(a2)    ; unit_VBlank eintragen
9C92          lea     78(a2),a1         ; a1 -> IOStdRequest im trackdisk.device
9C96          move.l  -20(a4),18(a1)    ; io_Unit eintragen
9C9C          move.l  -24(a4),14(a1)    ; io_Device eintragen
9CA2          move.l  #FE9EEE,74(a2)    ; is_Code eintragen
9CAA          move.l  a2,70(a2)         ; is_Data eintragen
9CAE          lea     62(a2),a0         ; a0 -> InterruptStruktur (ResetHandler)
9CB2          move.l  a0,28(a1)         ; a0 als io_Data eintragen
9CB6          move.b  #1,1E(a1)         ; io_Flags := QuickIO
9CBC          move.w  #B,1C(a1)         ; io_Command := kbd_AddResetHandler
9CC2          move.l  a6,-(a7)          ; a6 retten
9CC4          movea.l 14(a1),a6         ; a6 -> keyboard.device
9CC8          jsr     -1E(a6)           ; ---> BeginIO
9CCC          movea.l (a7)+,a6          ; a6 wiederherstellen
9CCE          move.w  #D,1C(a1)         ; io_Command := kbd_ResetHandlerDone
9CD4          move.b  #1,1E(a1)         ; io_Flags := QuickIO
9CDA          movea.l a2,a1             ; a1 := a2 -> trackdisk.device
9CDC          jsr     -1B0(a6)          ; ---> AddDevice
9CE0          moveq   #0,d0             ; d0 := 0 = ok-Flag
9CE2 FE9CE2   unlk    a4                ; Stackrahmen freigeben
9CE4          movea.l (a7)+,a2          ; a2 wiederherstellen
9CE6          rts
9CE8
9CE8 ;--------------------------------   Timer.device eröffnen
9CE8
9CE8 FE9CE8   move.l  d0,-(a7)          ; d0 retten
9CEA          lea     -1CB(pc),a0       ; a0 := FE9B21 -> 'timer.device'
```

```
9CEE          moveq   #0,d1                ; d1 := 0 = Flags
9CF0          jsr     -1BC(a6)             ; ---> OpenDevice
9CF4          tst.l   d0                   ; timer.device ok?
9CF6          beq.s   FE9D12               ; ja: ->
9CF8          movem.l d7/a5-a6,-(a7)       ; Register retten
9CFC          move.l  #14048015,d7         ; d7 := AlertKode
9D02          movea.l 4,a6                 ; a6 := SysBase
9D06          jsr     -6C(a6)              ; ---> Alert
9D0A          movem.l (a7)+,d7/a5-a6       ; Register wiederherstellen
9D0E          addq.l  #4,a7                ; Erzeugt Stackfehler!!
9D10          moveq   #1,d0                ; d0 := 1
9D12 FE9D12   addq.l  #4,a7                ; Stackzeiger korrigieren
9D14          rts
9D16
9D16 ;--------------------------------  MemoryEntry-Struktur
9D16
9D16 FE9D16   DC.L    0                   ; me_Node
9D1A          DC.L    0                   ; :
9D1E          DC.B    0                   ; :
9D1F          DC.B    0                   ; :
9D20          DC.L    0                   ; :
9D24          DC.W    3                   ; me_NumEntries: Zahl der Einträge
9D26          DC.L    10001               ; Attribute PUBLIC, CLEAR
9D2A          DC.L    19C                 ; 412 Bytes (trackdisk.unit)
9D2E          DC.L    10000               ; Attribut CLEAR
9D32          DC.L    200                 ; 512 Bytes (TaskStack)
9D36          DC.L    10003               ; Attribute PUBLIC, CHIP, CLEAR
9D3A          DC.L    4004                ; 16388 Bytes (Spurpuffer)
9D3E
9D3E ;--------------------------------  Unit erzeugen
9D3E
9D3E FE9D3E   movem.l d3-d4/a2-a4,-(a7)    ; Register retten
9D42          move.l  d1,d3                ; d3 := d1 = Flags
9D44          move.l  d2,d0                ; d0 := d2 = UnitNum
9D46          move.l  a6,-(a7)             ; a6 retten
9D48          movea.l 3C(a6),a6            ; a6 -> disk.resource
9D4C          jsr     -1E(a6)              ; ---> dr_GetUnitID
9D50          movea.l (a7)+,a6             ; a6 wiederherstellen
9D52          cmpi.l  #0,d0                ; DriveTyp = drt_Amiga (3.5")?
9D58          beq.s   FE9D6E               ; ja: ->
```

```
9D5A          cmpi.l  #55555555,d0        ; DriveTyp = drt_37422D2S (5.25")?
9D60          bne.s   FE9D68              ; nein: Fehler ->
9D62          btst.l  #0,d3               ; Flagbit 0 gesetzt?
9D66          bne.s   FE9D6E              ; ja: 5.25"-Laufwerk zugelassen ->
9D68 FE9D68   moveq   #21,d0              ; d0 := tderr_BadDriveType
9D6A          bra     FE9ED6              ; ---> Abschluß
9D6E
9D6E FE9D6E   move.l  d0,d4               ; d4 := d0 = DriveTyp
9D70          move.l  d2,d0               ; d0 := d2 = UnitNum
9D72          move.l  a6,-(a7)            ; a6 retten
9D74          movea.l 3C(a6),a6           ; a6 -> disk.resource
9D78          jsr     -6(a6)              ; ---> dr_AllocUnit
9D7C          movea.l (a7)+,a6            ; a6 wiederherstellen
9D7E          tst.l   d0                  ; erfolgreich?
9D80          bne.s   FE9D88              ; ja: ->
9D82          moveq   #22,d0              ; d0 := tderr_DriveInUse
9D84          bra     FE9ED6              ; ---> Abschluß
9D88
9D88 FE9D88   lea     -74(pc),a0          ; a0 := FE9D16 -> MemEntry-Struktur
9D8C          move.l  a6,-(a7)            ; a6 retten
9D8E          movea.l 34(a6),a6           ; a6 := SysBase
9D92          jsr     -DE(a6)             ; ---> AllocEntry
9D96          movea.l (a7)+,a6            ; a6 wiederherstellen
9D98          tst.l   d0                  ; Speicherbereiche zugeordnet?
9D9A          beq     FE9EDC              ; nein: Fehler ->
9D9E          movea.l d0,a4               ; a4 := d0 -> ME-Struktur
9DA0          movea.l 10(a4),a3           ; a3 -> trackdisk.unit
9DA4          lea     474(pc),a1          ; a1 := FEA21A -> InitStruct-Tabelle
9DA8          movea.l a3,a2               ; a2 := a3 -> trackdisk.unit
9DAA          move.w  #19C,d0             ; d0 := 412 Bytes
9DAE          move.l  a6,-(a7)            ; a6 retten
9DB0          movea.l 34(a6),a6           ; a6 := SysBase
9DB4          jsr     -4E(a6)             ; ---> InitStruct
9DB8          movea.l (a7)+,a6            ; a6 wiederherstellen
9DBA          move.l  a4,18A(a3)          ; Zeiger auf ME-Struktur eintragen
9DBE          move.b  d2,43(a3)           ; UnitNum eintragen
9DC2          move.b  #3,d0               ; d0 := 3
9DC6          add.b   d2,d0               ; + UnitNum ergibt SelectBit-Nummer
9DC8          moveq   #FF,d1              ; Alle Bits in d1 setzen
9DCA          bclr.l  d0,d1               ; SelectBit für Unit := 0
```

```
9DCC          move.b  d1,41(a3)          ; Kontrollbyte eintragen
9DD0          move.l  d4,3C(a3)          ; Laufwerk-Typ eintragen
9DD4          cmpi.l  #0,d4              ; 3.5"-Laufwerk?
9DDA          beq.s   FE9DE2             ; ja: ->
9DDC          lea     4F0(pc),a0         ; a0 := FEA2CE -> Parametertabelle 2
9DE0          bra.s   FE9DE6             ; --->
9DE2
9DE2 FE9DE2   lea     4D4(pc),a0         ; a0 := FEA2B8 -> Parametertabelle 1
9DE6 FE9DE6   lea     26(a3),a1          ; a1 -> Parameterbereich
9DEA          moveq   #16,d0             ; d0 := 22 = Zähleranfangswert
9DEC          bsr     FEA1EC             ; ---> Tabelle kopieren
9DF0          movea.l 18(a4),a0          ; a0 -> TaskStack (untere Grenze)
9DF4          move.l  a0,168(a3)         ; tc_SPLower := a0
9DF8          lea     200(a0),a0         ; a0 -> TaskStack (obere Grenze)
9DFC          move.l  a0,16C(a3)         ; tc_SPUpper := a0
9E00          move.l  a6,-(a0)           ; SysBase in Task Stack
9E02          move.l  a3,-(a0)           ; trackdisk.unit in Task Stack
9E04          move.l  a0,164(a3)         ; tc_SPReg := a0
9E08          movea.l 20(a4),a1          ; a1 -> Spurpuffer
9E0C          move.w  #-1,0(a1)          ; Spurnummer ungültig machen
9E12          clr.b   2(a1)              ; Änderungsflag löschen
9E16          move.l  a1,52(a3)          ; Pufferanfang in td.unit eintragen
9E1A          lea     4(a1),a0           ; a0 -> Pufferanfang+4
9E1E          move.w  #FFF,d0            ; d0 := 4095 = Zähleranfangswert
9E22          move.l  #AAAAAAAA,d1       ; Langwort mit Bitmuster 10101010...
9E28 FE9E28   move.l  d1,(a0)+           ; in Puffer kopieren
9E2A          dbra    d0,FE9E28          ; 4095 mal wiederholen ->
9E2E          lea     AE(a3),a0          ; a0 -> ReplyPort
9E32          move.l  a0,6C(a3)          ; a0 in IORequest unit_MicroHz eintragen
9E36          move.l  a0,DE(a3)          ; a0 in Message eintragen
9E3A          move.l  a3,F2(a3)          ; a3 in Interrupt-Struktur eintragen
9E3E          move.l  a3,94(a3)          ; a3 in IORequest unit_VBlank eintragen
9E42          lea     12E(a3),a0         ; a0 -> Task-Kontrollblock
9E46          move.l  a0,BE(a3)          ; a0 als mp_SigTask in ReplyPort eintr.
9E4A          move.l  4E(a6),72(a3)      ; timer.device in td.unit eintragen
9E50          move.l  52(a6),76(a3)      ; unit_MicroHz in td.unit eintragen
9E56          move.l  56(a6),9A(a3)      ; timer.device in td.unit eintragen
9E5C          move.l  5A(a6),9E(a3)      ; unit_VBlank in td.unit eintragen
9E62          lea     18E(a3),a0         ; a0 -> ChangeIntList
9E66          move.l  a0,(a0)            ; Macro NewList
```

```
9E68          addq.l  #4,(a0)             ; :
9E6A          clr.l   4(a0)               ; :
9E6E          move.l  a0,8(a0)            ; :
9E72          lea     14(a3),a0           ; a0 -> mp_MsgList im MsgPort
9E76          move.l  a0,(a0)             ; Macro NewList
9E78          addq.l  #4,(a0)             ; :
9E7A          clr.l   4(a0)               ; :
9E7E          move.l  a0,8(a0)            ; :
9E82          lea     C2(a3),a0           ; a0 -> mp_MsgList im ReplyPort
9E86          move.l  a0,(a0)             ; Macro NewList
9E88          addq.l  #4,(a0)             ; :
9E8A          clr.l   4(a0)               ; :
9E8E          move.l  a0,8(a0)            ; :
9E92          lea     178(a3),a0          ; a0 -> tc_MemEntry
9E96          move.l  a0,(a0)             ; Macro NewList
9E98          addq.l  #4,(a0)             ; :
9E9A          clr.l   4(a0)               ; :
9E9E          move.l  a0,8(a0)            ; :
9EA2          movea.l 18A(a3),a1          ; a1 -> MemEntry-Struktur
9EA6          move.l  (a0),d0             ; Macro AddHead
9EA8          move.l  a1,(a0)             ; :
9EAA          movem.l d0/a0,(a1)          ; :
9EAE          movea.l d0,a0               ; :
9EB0          move.l  a1,4(a0)            ; :
9EB4          movea.l a3,a4               ; a3 in a4 retten
9EB6          lea     12E(a3),a1          ; a1 -> Task-Kontrollblock
9EBA          lea     F94(pc),a2          ; a2 := FEAE50 = InitialPC (Task-Anfang)
9EBE          lea     FFFFFFFF,a3         ; a3 := FinalPC = -1 (Ausgang Dispatch)
9EC4          moveq   #0,d0               ; d0 := 0
9EC6          move.l  a6,-(a7)            ; a6 retten
9EC8          movea.l 34(a6),a6           ; a6 := SysBase
9ECC          jsr     -11A(a6)            ; ---> AddTask
9ED0          movea.l (a7)+,a6            ; a6 wiederherstellen
9ED2          moveq   #0,d0               ; d0 := 0
9ED4          movea.l a4,a0               ; a0 := a4 -> MsgPort
9ED6 FE9ED6   movem.l (a7)+,d3-d4/a2-a4   ; Register wiederherstellen
9EDA          rts
9EDC
9EDC ;------ Fehlerausgang bei Speichermangel
9EDC
```

```
9EDC FE9EDC  move.l  d2,dØ              ; dØ := d2
9EDE         move.l  a6,-(a7)           ; a6 retten
9EEØ         movea.l 3C(a6),a6          ; a6 -> disk.resource
9EE4         jsr     -C(a6)             ; ---> dr_FreeUnit
9EE8         movea.l (a7)+,a6           ; a6 wiederherstellen
9EEA         moveq   #1F,dØ             ; dØ := tderr_NoMem
9EEC         bra.s   FE9ED6             ; ---> Abschluß
9EEE
9EEE ;-------------------------------   Reset-Handler
9EEE
9EEE FE9EEE  move.l  a2,-(a7)           ; a2 retten
9EFØ         movea.l a1,a2              ; a2 := a1 -> trackdisk.device
9EF2         bset.b  #4,22(a1)          ; Reset-Flag setzen
9EF8         btst.b  #5,22(a1)          ; Schreibflag gesetzt?
9EFE         bne.s   FE9F1Ø             ; ja: ->
9FØØ         lea     78(a2),a1          ; a1 -> IORequest Keyboard
9FØ4         move.l  a6,-(a7)           ; a6 retten
9FØ6         movea.l 14(a1),a6          ; a6 -> io_Device = keyboard.device
9FØA         jsr     -1E(a6)            ; ---> BeginIO: ResetHandlerDone
9FØE         movea.l (a7)+,a6           ; a6 wiederherstellen
9F1Ø FE9F1Ø  movea.l (a7)+,a2           ; a2 wiederherstellen
9F12         rts
9F14
9F14 ;-------------------------------   Expunge
9F14
9F14 ; Diese Routine tut überhaupt nichts und könnte durch ihre letzten
9F14 ; beiden Zeilen ersetzt werden.
9F14
9F14 FE9F14  movea.l 4,aØ               ; aØ := SysBase
9F18         move.w  #4ØØØ,DFFØ9A       ; Interrupts sperren
9F2Ø         addq.b  #1,126(aØ)         ; IDNestCnt erhöhen
9F24         tst.w   2Ø(a6)             ; Openzähler auf Ø?
9F28         bne     FE9F2C             ; nein: -> (Sprungziel!!)
9F2C FE9F2C  movea.l 4,aØ               ; aØ := SysBase
9F3Ø         subq.b  #1,126(aØ)         ; IDNestCnt erniedrigen
9F34         bge.s   FE9F3E             ; Interrupts noch gesperrt: ->
9F36         move.w  #CØØØ,DFFØ9A       ; Interrupts freigeben
9F3E FE9F3E  moveq   #Ø,dØ              ; dØ := Ø
9F4Ø         rts
9F42
```

```
9F42 ;--------------------------------  Open
9F42
9F42 FE9F42  movem.l d2/a2/a4,-(a7)      ; Register retten
9F46         movea.l a1,a4               ; a4 := a1 -> IORequest
9F48         move.l  d0,d2               ; d2 := d0 = UnitNum
9F4A         cmpi.l  #4,d0               ; d0 < 4?
9F50         bcs.s   FE9F56              ; ja: ->
9F52         moveq   #20,d0              ; d0 := tderr_BadUnitNum
9F54         bra.s   FE9F82              ; ---> Abschluß
9F56
9F56 FE9F56  lsl.w   #2,d0               ; d0 * 4 ergibt Offset
9F58         lea     24(a6),a2           ; a2 -> UnitPointer-Bereich
9F5C         adda.l  d0,a2               ; a2 -> UnitPointer
9F5E         movea.l (a2),a0             ; a0 := UnitPointer
9F60         move.l  a0,d0               ; d0 := a0 = UnitPointer
9F62         bne.s   FE9F70              ; UnitPointer vorhanden: ->
9F64         bsr     FE9D3E              ; ---> Unit erzeugen
9F68         tst.l   d0                  ; Unit ok?
9F6A         bne     FE9F82              ; nein: Abschluß ->
9F6E         move.l  a0,(a2)             ; UnitPointer eintragen
9F70 FE9F70  move.l  a0,18(a4)           ; io_Unit := UnitPointer
9F74         addq.w  #1,20(a6)           ; Device-Openzähler erhöhen
9F78         addq.w  #1,24(a0)           ; Unit-Openzähler erhöhen
9F7C FE9F7C  movem.l (a7)+,d2/a2/a4      ; Register wiederherstellen
9F80         rts
9F82
9F82 FE9F82  move.b  d0,1F(a4)           ; io_Error := Fehlerkode
9F86         moveq   #-1,d0              ; d0 := -1
9F88         move.l  d0,18(a4)           ; io_Unit := -1
9F8C         move.l  d0,14(a4)           ; io_Device := -1
9F90         bra.s   FE9F7C              ; ---> Abschluß
9F92
9F92 ;--------------------------------  Close
9F92
9F92 FE9F92  movem.l a2-a3,-(a7)         ; Register retten
9F96         movea.l a1,a2               ; a2 := a1 -> IORequest
9F98         movea.l 18(a2),a3           ; a3 := io_Unit
9F9C         subq.w  #1,24(a3)           ; Unit-Openzähler erniedrigen
9FA0         bne.s   FE9FA8              ; noch nicht 0: ->
9FA2         bset.b  #3,40(a3)           ; Close-Flag setzen
```

```
9FA8 FE9FA8  subq.w  #1,2Ø(a6)          ; Device-Openzähler erniedrigen
9FAC         moveq   #-1,dØ             ; dØ := -1
9FAE         move.l  dØ,18(a2)          ; io_Unit := -1
9FB2         move.l  dØ,14(a2)          ; io_Device := -1
9FB6         movem.l (a7)+,a2-a3        ; Register wiederherstellen
9FBA         moveq   #Ø,dØ              ; dØ := Ø
9FBC         rts
9FBE
9FBE ;-------------------------------   BeginIO
9FBE
9FBE FE9FBE  clr.b   1F(a1)             ; io_Error löschen
9FC2         moveq   #Ø,dØ              ; dØ := Ø
9FC4         move.b  1D(a1),dØ          ; dØ := io_Command
9FC8         cmpi.b  #16,dØ             ; dØ > 22?
9FCC         bcc.s   FEAØ16             ; ja: Fehler ->
9FCE         movea.l 18(a1),aØ          ; aØ := io_Unit
9FD2         move.l  #ØC61C2,d1         ; d1 := %11ØØØ11ØØØØ111ØØØØ1Ø
9FD8         btst.l  dØ,d1              ; Bit zur Kommandonummer testen
9FDA         bne.s   FE9FFØ             ; Bit gesetzt: ->
9FDC         andi.b  #7E,1E(a1)         ; io_Flags: QuickIO und Bit 7 löschen
9FE2         move.l  a6,-(a7)           ; a6 retten
9FE4         movea.l 34(a6),a6          ; a6 := SysBase
9FE8         jsr     -16E(a6)           ; ---> PutMsg
9FEC         movea.l (a7)+,a6           ; a6 wiederherstellen
9FEE         bra.s   FEAØ14             ; ---> rts
9FFØ
9FFØ ; Dieser Teil wird durchlaufen bei den Kommandos 1,6,7,8,13,14,18,19
9FFØ
9FFØ FE9FFØ  bset.b  #7,1E(a1)          ; io_Flags: NoSignal-Flag setzen
9FF6         move.b  #5,8(a1)           ; ln_Type := nt_Message
9FFC         movem.l a2-a3,-(a7)        ; Register retten
AØØØ         movea.l aØ,a3              ; a3 := aØ = io_Unit
AØØ2         movea.l a1,a2              ; a2 := a1 -> IORequest
AØØ4         lea     2FA(pc),aØ         ; aØ := FEA3ØØ -> Kommandotabelle
AØØ8         lsl.w   #2,dØ              ; io_Command mal 4 = Offset in Tabelle
AØØA         movea.l Ø(aØ,dØ.w),aØ      ; aØ := Routinenadresse
AØØE         jsr     (aØ)               ; ---> Kommando ausführen
AØ1Ø         movem.l (a7)+,a2-a3        ; Register wiederherstellen
AØ14 FEAØ14  rts
AØ16
```

```
A016 FEA016  bsr     FEA06E              ; ---> Kommando-Fehler melden
A01A         bra.s   FEA014              ; ---> rts
A01C
A01C ;-------------------------------   Kommandoverteiler
A01C
A01C FEA01C  move.l  a2,-(a7)            ; a2 retten
A01E         movea.l a1,a2               ; a1 in a2 retten
A020         andi.b  #FA,40(a3)          ; ExtCmd- und SecLabel-Flag löschen
A026         bsr     FE998C              ; ---> Prüfen ob Disk im Laufwerk
A02A         movea.l a2,a1               ; a1 wiederherstellen
A02C         move.w  1C(a2),d0           ; d0 := io_Command
A030         btst.l  #F,d0               ; = Extended Command?
A034         beq.s   FEA052              ; nein: ->
A036         bset.b  #2,40(a3)           ; ExtCmd-Flag setzen
A03C         move.l  126(a3),d1          ; d1 := Diskwechsel-Zählerstand
A040         cmp.l   30(a2),d1           ; > iotd_Count im IORequest?
A044         bls.s   FEA052              ; nein: ->
A046         move.b  #1D,1F(a2)          ; io_Error := tderr_DiskChanged
A04C         bsr     FEA1B0              ; ---> IORequest beantworten
A050         bra.s   FEA066              ; ---> Abschluß
A052
A052 FEA052  moveq   #0,d1               ; d1 := 0
A054         move.b  d0,d1               ; d1 := io_Command
A056         lsl.w   #2,d1               ; mal 4 ergibt Tabellenoffset
A058         lea     2A6(pc),a0          ; a0 := FEA300 -> Kommandotabelle
A05C         movea.l 0(a0,d1.w),a0       ; a0 := Routinenadresse
A060         jsr     (a0)                ; ---> Kommando ausführen
A062         bsr     FE998C              ; ---> Prüfen ob Disk im Laufwerk
A066 FEA066  movea.l (a7)+,a2            ; a2 wiederherstellen
A068         rts
A06A
A06A ;-------------------------------   Extfunct, AbortIO
A06A
A06A FEA06A  moveq   #0,d0               ; d0 := 0
A06C         rts
A06E
A06E ;-------------------------------   Kommando-Fehlerausgang
A06E
A06E FEA06E  move.b  #FD,1F(a1)          ; io_Error := ioerr_NoCmd
A074         bsr     FEA1B0              ; ---> IORequest beantworten
```

```
A078          rts
A07A
A07A ;--------------------------------   td_Format
A07A
A07A FEA07A   movem.l d2-d5/a2-a5,-(a7)   ; Register retten
A07E          movea.l a1,a4               ; a4 := a1 -> IORequest
A080          movea.l 18(a4),a3           ; a3 := io_Unit
A084          moveq   #1,d0               ; d0 := 1
A086          bsr     FEA462              ; ---> Motor einschalten
A08A          tst.l   d0                  ; Motor läuft?
A08C          bne.s   FEA098              ; ja: ->
A08E          move.l  #0493E0,d0          ; d0 := 300000
A094          bsr     FEA4F0              ; ---> d0 * 2 us warten
A098 FEA098   movea.l 52(a3),a0           ; a0 -> Spurpuffer
A09C          move.l  a0,4E(a3)           ; Schreib-Datenbereich := a0
A0A0          bclr.b  #0,2(a0)            ; Änderungsflag := 0
A0A6          beq.s   FEA0BA              ; war nicht gesetzt: ->
A0A8          bsr     FEA958              ; ---> Spur auf Disk schreiben
A0AC          tst.l   d0                  ; ok?
A0AE          beq     FEA0BA              ; ja: ->
A0B2          move.b  d0,1F(a4)           ; io_Error := d0
A0B6          bra     FEA16E              ; ---> Abschluß
A0BA
A0BA FEA0BA   move.l  2C(a4),d0           ; d0 := io_Offset
A0BE          bsr     FEA182              ; ---> Spurzahl für d0 Bytes ermitteln
A0C2          move.l  d0,d2               ; d2 := d0 = Anfangsspur
A0C4          tst.l   d1                  ; Restsektoren vorhanden?
A0C6          bne     FEA17A              ; ja: Fehler ->
A0CA          move.l  24(a4),d0           ; d0 := io_Length
A0CE          bsr     FEA182              ; ---> Spurzahl für d0 Bytes ermitteln
A0D2          tst.l   d1                  ; Restsektoren vorhanden?
A0D4          bne     FEA17A              ; ja: Fehler ->
A0D8          move.l  d0,d3               ; d3 := d0 = Spurzahl
A0DA          beq     FEA16E              ; = 0: Fertig ->
A0DE          add.l   d2,d0               ; d0 := Anfangsspur+Spurzahl
A0E0          cmp.l   36(a3),d0           ; d0 > Zahl der Spuren pro Disk?
A0E4          bhi     FEA17A              ; ja: Fehler ->
A0E8          movea.l 28(a4),a5           ; a5 := io_Data
A0EC FEA0EC   move.l  d2,d0               ; d0 := Anfangsspur
A0EE          bsr     FEA3DA              ; ---> Kopf auf Spur d0 setzen
```

```
A0F2            movea.l 52(a3),a2           ; a2 -> Spurpuffer
A0F6            lea     4(a2),a2            ; a2 -> Datenbereich im Puffer
A0FA            move.w  #FFF,d0             ; d0 := 4095 = Zähler-Anfangswert
A0FE            move.l  #AAAAAAAA,d1        ; d1 := Bitmuster 10101010...
A104 FEA104     move.l  d1,(a2)+            ; Bitmuster in Puffer kopieren
A106            dbra    d0,FEA104           ; bis 4096 Langworte kopiert ->
A10A            movea.l 52(a3),a2           ; a2 -> Spurpuffer
A10E            lea     680(a2),a2          ; a2 -> Spurpuffer + 1664 (hinter Lücke)
A112            moveq   #B,d4               ; d4 := 11 = Sektoroffset zur Lücke
A114            moveq   #0,d5               ; d5 := 0 = Nummer des 1. Sektors
A116 FEA116     move.l  #FF000000,d0        ; d0 ergibt Header-Langwort
A11C            move.l  d5,d1               ; d1 := d5 = Sektornummer
A11E            lsl.l   #8,d1               ; um 1 Byte linksverschieben
A120            or.l    d1,d0               ; und in d0 einodern
A122            or.l    d4,d0               ; Offset zur Lücke einodern
A124            move.l  d2,d1               ; d1 := d2 = Spurnummer
A126            swap    d1                  ; ins obere Wort setzen
A128            or.l    d1,d0               ; und in d0 einodern
A12A            movea.l a2,a1               ; a1 := a2 -> Spurpuffer hinter Lücke
A12C            movea.l a5,a0               ; a0 := a5 -> Sektordaten
A12E            bsr     FEAADC              ; ---> Sektordaten im Puffer erzeugen
A132            addq.l  #1,d5               ; Sektornummer erhöhen
A134            adda.l  #440,a2             ; Pufferzeiger um 1088 erhöhen
A13A            adda.l  #200,a5             ; Datenzeiger um 512 erhöhen
A140            subq.l  #1,d4               ; Sektoroffset zur Lücke erniedrigen
A142            bne.s   FEA116              ; noch nicht 0: Loop ->
A144            movea.l 52(a3),a0           ; a0 -> Spurpuffer
A148            lea     4(a0),a0            ; a0 := Anfang des Datenbereichs
A14C            move.w  #353E,d0            ; d0 := 13630 = 1660 + 11*1088 + 2
A150            bsr     FEA5B4              ; ---> d0 Bytes ab Adresse a0 schreiben
A154            tst.l   d0                  ; ok?
A156            beq.s   FEA15E              ; ja: ->
A158            move.b  d0,1F(a4)           ; io_Error := d0
A15C            bra.s   FEA16E              ; ---> Abschluß
A15E
A15E FEA15E     addq.l  #1,d2               ; Spurnummer erhöhen
A160            subq.l  #1,d3               ; Spurzähler erniedrigen
A162            bne.s   FEA0EC              ; noch nicht 0: Loop ->
A164            movea.l 52(a3),a2           ; a2 -> Spurpuffer
A168            move.w  #-1,0(a2)           ; Spurnummer im Puffer ungültig machen
```

```
A16E FEA16E  movea.l a4,a1              ; a1 := a4 -> IORequest
A170         bsr     FEA1B0             ; ---> IORequest beantworten
A174         movem.l (a7)+,d2-d5/a2-a5  ; Register wiederherstellen
A178         rts
A17A
A17A FEA17A  move.b  #FC,1F(a4)         ; io_Error := ioerr_BadLength
A180         bra.s   FEA16E             ; ---> Abschluß
A182
A182 ;------ Zahl der Bytes in Spuren und Sektoren umrechnen
A182
A182 FEA182  move.l  d0,d1              ; d1 := d0 = Bytezahl
A184         andi.l  #1FF,d1            ; modulo 512
A18A         bne     FEA1AA             ; nicht 0: Fehler ->
A18E         cmp.l   38(a3),d0          ; d0 > Zahl der Bytes pro Disk?
A192         bhi.s   FEA1AA             ; ja: Fehler ->
A194         moveq   #9,d1              ; d1 := 9 = ld 512
A196         lsr.l   d1,d0              ; d0 := d0 / 512 ergibt Blockzahl
A198         divu    #B,d0              ; / 11 ergibt Zahl der Spuren
A19C         moveq   #0,d1              ; d1 := 0
A19E         swap    d0                 ; d0.w := Zahl der Restsektoren
A1A0         move.w  d0,d1              ; d1 := d0.w = Zahl der Restsektoren
A1A2         move.w  #0,d0              ; d0.w := 0
A1A6         swap    d0                 ; d0 := Spurzahl
A1A8         rts
A1AA
A1AA FEA1AA  moveq   #-1,d0             ; Fehlerausgang
A1AC         moveq   #-1,d1
A1AE         rts
A1B0
A1B0 ;-------------------------------   IORequest beantworten
A1B0
A1B0 FEA1B0  move.l  a3,-(a7)           ; a3 retten
A1B2         movea.l 18(a1),a3          ; a3 := io_Unit
A1B6         btst.b  #0,1E(a1)          ; QuickIO gesetzt?
A1BC         bne.s   FEA1E8             ; ja: ->
A1BE         move.l  a6,-(a7)           ; a6 retten
A1C0         movea.l 34(a6),a6          ; a6 := SysBase
A1C4         jsr     -17A(a6)           ; ---> ReplyMsg
A1C8         movea.l (a7)+,a6           ; a6 wiederherstellen
A1CA         btst.b  #7,1E(a1)          ; io_Flags: NoSignal-Flag gesetzt?
```

```
A1D0          bne.s   FEA1E8              ; ja: ->
A1D2          move.l  #100,d0             ; d0 := Signal
A1D8          lea     12E(a3),a1          ; a1 -> Task
A1DC          move.l  a6,-(a7)            ; a6 retten
A1DE          movea.l 34(a6),a6           ; a6 := SysBase
A1E2          jsr     -144(a6)            ; ---> Signal
A1E6          movea.l (a7)+,a6            ; a6 wiederherstellen
A1E8 FEA1E8   movea.l (a7)+,a3            ; a3 wiederherstellen
A1EA          rts
A1EC
A1EC ;--------------------------------   Kopierroutine
A1EC
A1EC FEA1EC   bra.s   FEA1F0              ; ---> Kopierschleife
A1EE
A1EE FEA1EE   move.b  (a0)+,(a1)+         ; Byte kopieren
A1F0 FEA1F0   dbra    d0,FEA1EE           ; wiederholen bis d0 = -1 ->
A1F4          rts
A1F6
A1F6 ;--------------------------------   Tabellen für InitStruct
A1F6
A1F6 ;------ Initialisierung von trackdisk.device
A1F6
A1F6 FEA1F6   DC.W    E000                ; InitByte
A1F8          DC.W    8                   ; ln_Type
A1FA          DC.B    3                   ; = nt_Device
A1FB          DC.B    0
A1FC          DC.W    C000                ; InitLong
A1FE          DC.W    A                   ; ln_Name
A200          DC.L    FE98FE              ; -> 'trackdisk.device'
A204          DC.W    E000                ; InitByte
A206          DC.W    E                   ; lib_Flags
A208          DC.B    6                   ; = Changed, Sumused
A209          DC.B    0
A20A          DC.W    D000                ; InitWord
A20C          DC.W    14                  ; lib_Version
A20E          DC.W    21                  ; = 33
A210          DC.W    D000                ; InitWord
A212          DC.W    16                  ; lib_Revision
A214          DC.W    7F                  ; = 127
A216          DC.W    0
```

```
A218          DC.W    Ø
A21A
A21A ;------ Initialisierung von trackdisk.unit
A21A
A21A FEA21A   DC.W    EØØØ                ; InitByte
A21C          DC.W    8                   ; ln_Type
A21E          DC.B    4                   ; = nt_MsgPort
A21F          DC.B    Ø
A22Ø          DC.W    CØØØ                ; InitLong
A222          DC.W    A                   ; ln_Name
A224          DC.L    FE98FE              ; -> 'trackdisk.device'
A228          DC.W    EØØØ                ; InitByte
A22A          DC.W    F                   ; mp_SigBit
A22C          DC.B    9                   ; = Bitnummer
A22D          DC.B    Ø
A22E          DC.W    EØØØ                ; InitByte
A23Ø          DC.W    2Ø                  ; lh_Type
A232          DC.B    5                   ; = nt_Message
A233          DC.B    Ø
A234          DC.W    DØØØ                ; InitWord
A236          DC.W    4C                  ; Spurnummer
A238          DC.W    -1                  ; 'ungültig'
A23A          DC.W    EØØØ                ; InitByte
A23C          DC.W    4Ø                  ; Flags
A23E          DC.B    2                   ; 'NoDisk'
A23F          DC.B    Ø
A24Ø          DC.W    EØØØ                ; InitByte
A242          DC.W    B6                  ; ln_Type im ReplyPort
A244          DC.B    4                   ; = nt_MsgPort
A245          DC.B    Ø
A246          DC.W    CØØØ                ; InitLong
A248          DC.W    B8                  ; ln_Name im ReplyPort
A24A          DC.L    FE98FE              ; -> 'trackdisk.device'
A24E          DC.W    EØØØ                ; InitByte
A25Ø          DC.W    BD                  ; mp_SigBit im ReplyPort
A252          DC.B    A                   ; = 1Ø
A253          DC.B    Ø
A254          DC.W    EØØØ                ; InitByte
A256          DC.W    CE                  ; lh_Type im ReplyPort, Message List
A258          DC.B    5                   ; = nt_Message
```

```
A259        DC.B    Ø
A25A        DC.W    EØØØ            ; InitByte
A25C        DC.W    66              ; ln_Type in IORequest unit_MicroHz
A25E        DC.B    5               ; = nt_Message
A25F        DC.B    Ø
A26Ø        DC.W    CØØØ            ; InitLong
A262        DC.W    68              ; ln_Name in IORequest unit_MicroHz
A264        DC.L    FE98FE          ; -> 'trackdisk.device'
A268        DC.W    DØØØ            ; InitWord
A26A        DC.W    7A              ; io_Command in IORequest unit_MicroHz
A26C        DC.W    9               ; = tr_AddRequest
A26E        DC.W    EØØØ            ; InitByte
A27Ø        DC.W    8E              ; ln_Type in IORequest unit_VBlank
A272        DC.B    5               ; = nt_Message
A273        DC.B    Ø
A274        DC.W    CØØØ            ; InitLong
A276        DC.W    9Ø              ; ln_Name in IORequest unit_VBlank
A278        DC.L    FE98FE          ; -> 'trackdisk.device'
A27C        DC.W    DØØØ            ; InitWord
A27E        DC.W    A2              ; io_Command in IORequest unit_VBlank
A28Ø        DC.W    9               ; = tr_AddRequest
A282        DC.W    CØØØ            ; InitLong
A284        DC.W    A6              ; tr_Secs in IORequest unit_VBlank
A286        DC.L    A               ; = 1Ø
A28A        DC.W    EØØØ            ; InitByte
A28C        DC.W    136             ; ln_Type in trackdisk.task
A28E        DC.B    1               ; = nt_Task
A28F        DC.B    Ø
A29Ø        DC.W    EØØØ            ; InitByte
A292        DC.W    137             ; ln_Pri in trackdisk.task
A294        DC.B    5               ; = 5
A295        DC.B    Ø
A296        DC.W    CØØØ            ; InitLong
A298        DC.W    138             ; ln_Name in trackdisk.task
A29A        DC.L    FE98FE          ; -> 'trackdisk.device'
A29E        DC.W    EØØØ            ; InitByte
A2AØ        DC.W    D8              ; ln_Type
A2A2        DC.B    5               ; = nt_Message
A2A3        DC.B    Ø
A2A4        DC.W    CØØØ            ; InitLong
```

```
A2A6          DC.W    DA                ; ln_Name
A2A8          DC.L    FE98FE            ; -> 'trackdisk.device'
A2AC          DC.W    C000              ; InitLong
A2AE          DC.W    F6                ; is_Code
A2B0          DC.L    FEA6F2            ; -> Interrupt-Handler
A2B4          DC.W    0
A2B6          DC.W    0
A2B8
A2B8 ;--------------------------------   Laufwerk-Parameter für trackdisk.unit
A2B8
A2B8 ;------ Amiga-Laufwerk
A2B8
A2B8 FEA2B8   DC.W    50                ; tdu_Comp01Track = 80
A2BA          DC.W    FFFF              ; tdu_Comp10Track = 65535
A2BC          DC.W    FFFF              ; tdu_Comp11Track = 65535
A2BE          DC.L    BB8               ; tdu_StepDelay   = 3000
A2C2          DC.L    3A98              ; tdu_SettleDelay = 15000
A2C6          DC.B    A                 ; tdu_RetryCnt    = 10
A2C7          DC.B    1                 ; Typ Drive3_5
A2C8          DC.W    A0                ; Spuren pro Disk = 160
A2CA          DC.L    DC000             ; Bytes pro Disk  = 901120
A2CE
A2CE ;------ 5.25" Laufwerk
A2CE
A2CE FEA2CE   DC.W    28                ; tdu_Comp01Track = 40
A2D0          DC.W    FFFF              ; tdu_Comp10Track = 65535
A2D2          DC.W    FFFF              ; tdu_Comp11Track = 65535
A2D4          DC.L    1770              ; tdu_StepDelay   = 6000
A2D8          DC.L    3A98              ; tdu_SettleDelay = 15000
A2DC          DC.B    A                 ; tdu_RetryCnt    = 10
A2DD          DC.B    2                 ; Typ Drive5_25
A2DE          DC.W    50                ; Spuren pro Disk = 80
A2E0          DC.L    6E000             ; Bytes pro Disk  = 450560
A2E4
A2E4 ;--------------------------------   Vektortabelle für Sprungliste
A2E4
A2E4 FEA2E4   DC.L    FE9F42            ; Open
A2E8          DC.L    FE9F92            ; Close
A2EC          DC.L    FE9F14            ; Expunge
A2F0          DC.L    FEA06A            ; Extfunct
```

```
A2F4          DC.L    FE9FBE              ; BeginIO
A2F8          DC.L    FEA06A              ; AbortIO
A2FC          DC.L    -1
A300
A300 ;--------------------------------   Tabelle der Kommandoroutinen
A300
A300 FEA300   DC.L    FEA06E              ; $00  cmd_Invalid
A304          DC.L    FEA06E              ; $01  cmd_Reset
A308          DC.L    FEA734              ; $02  cmd_Read
A30C          DC.L    FEA734              ; $03  cmd_Write
A310          DC.L    FEAAAA              ; $04  cmd_Update
A314          DC.L    FEAA94              ; $05  cmd_Clear
A318          DC.L    FEA06E              ; $06  cmd_Stop
A31C          DC.L    FEA06E              ; $07  cmd_Start
A320          DC.L    FEA06E              ; $08  cmd_Flush
A324          DC.L    FEA9FE              ; $09  td_Motor
A328          DC.L    FEAA14              ; $0A  td_Seek
A32C          DC.L    FEA07A              ; $0B  td_Format
A330          DC.L    FE9AB6              ; $0C  td_Remove
A334          DC.L    FE9A96              ; $0D  td_ChangeNum
A338          DC.L    FE9AA2              ; $0E  td_ChangeState
A33C          DC.L    FEAA44              ; $0F  td_ProtStatus
A340          DC.L    FEB2E8              ; $10  td_RawRead
A344          DC.L    FEB2EE              ; $11  td_RawWrite
A348          DC.L    FEB3B6              ; $12  td_GetDriveType
A34C          DC.L    FEB3C8              ; $13  td_GetNumTracks
A350          DC.L    FE9AC2              ; $14  td_AddChangeInt
A354          DC.L    FE9ADE              ; $15  td_RemChangeInt
A358
A358 ;--------------------------------   Kopf auf Spur 0 setzen
A358
A358 FEA358   movem.l d2,-(a7)            ; d2 retten
A35C          bsr     FEADDC              ; ---> Laufwerk anfordern
A360          move.b  41(a3),d0           ; d0 := Kontrollbyte
A364          bset.l  #2,d0               ; DSKSIDE := 1 (unten)
A368          bclr.l  #1,d0               ; DSKDIREC := 0 (nach innen)
A36C          move.b  d0,BFD100           ; Kontrollbyte schreiben
A372          move.b  d0,41(a3)           ; Kontrollbyte speichern
A376          moveq   #0,d2               ; d2 := 0
A378          move.w  36(a3),d2           ; d2 := Zahl der Spuren pro Disk
```

```
A37C          lsr.l   #2,d2              ; durch 4 dividieren
A37E FEA37E   btst.b  #4,BFE001         ; DSKTRACK0: Kopf auf Spur 0?
A386          bne.s   FEA390            ; nein: ->
A388          bsr     FEA4BE            ; ---> Kopfschritt
A38C          dbra    d2,FEA37E         ; wiederholen bis d2 = -1 ->
A390 FEA390   bset.b  #1,41(a3)         ; DSKDIREC := 1 (nach aussen)
A396          clr.w   4C(a3)            ; Spurnummer := 0
A39A          move.w  36(a3),d2         ; d2 := Zahl der Spuren pro Disk
A39E          lsr.w   #1,d2             ; / 2 = Zahl der Spuren pro Seite
A3A0          addi.w  #14,d2            ; + 20 (zur Sicherheit)
A3A4          bra.s   FEA3AA            ; ---> Schrittschleife
A3A6
A3A6 FEA3A6   bsr     FEA4BE            ; ---> Kopfschritt
A3AA FEA3AA   btst.b  #4,BFE001         ; DSKTRACK0: Kopf auf Spur 0?
A3B2          dbeq    d2,FEA3A6         ; wiederholen bis 'ja' oder d2 = -1: ->
A3B6          move.b  BFE001,d2         ; d2 := Diskstatus
A3BC          bsr     FEAE42            ; ---> Laufwerk freigeben
A3C0          btst.l  #4,d2             ; DSKTRACK0: Kopf auf Spur 0?
A3C4          beq.s   FEA3CA            ; ja: ->
A3C6          moveq   #1E,d0            ; d0 := tderr_SeekError
A3C8          bra.s   FEA3D4            ; ---> Abschluß
A3CA
A3CA FEA3CA   move.l  30(a3),d0         ; d0 := tdu_SettleDelay
A3CE          bsr     FEA4F0            ; ---> d0 * 2 us warten
A3D2          moveq   #0,d0             ; d0 := 0
A3D4 FEA3D4   movem.l (a7)+,d2          ; d2 wiederherstellen
A3D8          rts
A3DA
A3DA ;--------------------------------   Kopf auf Spur d0 setzen
A3DA
A3DA FEA3DA   movem.l d2-d3,-(a7)       ; Register retten
A3DE          move.w  4C(a3),d2         ; d2 := Nummer der Spur unter Kopf
A3E2          bpl     FEA3F4            ; gültige Spurnummer: ->
A3E6          move.l  d0,d2             ; d2 := d0 = gesuchte Spur
A3E8          bsr     FEA358            ; ---> Kopf auf Spur 0 setzen
A3EC          tst.l   d0                ; Erfolg gehabt?
A3EE          bne     FEA45C            ; nein: ->
A3F2          exg     d2,d0             ; d0 wiederherstellen, d2 = 0
A3F4 FEA3F4   cmp.w   d0,d2             ; gesuchte Spur = Spur unter Kopf?
A3F6          beq     FEA45A            ; ja: Fertig->
```

```
A3FA          move.l  d0,d3               ; d3 := d0 = gesuchte Spur
A3FC          bsr     FEADDC              ; ---> Laufwerk anfordern
A400          move.b  41(a3),d1           ; d1 := Kontrollbyte
A404          bset.l  #2,d1               ; DSKSIDE := 1 (unten)
A408          btst.l  #0,d3               ; gesuchte Spurnummer gerade?
A40C          beq.s   FEA412              ; ja: ->
A40E          bclr.l  #2,d1               ; DSKSIDE := 0 (oben)
A412 FEA412   move.b  d1,41(a3)           ; Kontrollbyte speichern
A416          move.b  d1,BFD100           ; Kontrollbyte schreiben
A41C          move.w  d3,4C(a3)           ; Nummer der neuen Spur unter Kopf := d3
A420          lsr.w   #1,d2               ; d2 halbieren
A422          lsr.w   #1,d3               ; d3 halbieren
A424          sub.w   d3,d2               ; d2 := d2-d3 = Kopfschritt-Vektor
A426          bcs.s   FEA430              ; negativ: nach innen ->
A428          bhi.s   FEA43C              ; positiv: nach außen ->
A42A          bsr     FEAE42              ; ---> Laufwerk freigeben
A42E          bra.s   FEA45A              ; ---> Abschluß
A430
A430 FEA430   bclr.l  #1,d1               ; DSKDIREC := 0 (nach innen)
A434          move.b  d1,41(a3)           ; Kontrollbyte schreiben
A438          neg.w   d2                  ; Schrittzahl positiv machen
A43A          bra.s   FEA44A              ; ---> Schrittschleife
A43C
A43C FEA43C   bset.l  #1,d1               ; DSKDIREC := 1 (nach außen)
A440          move.b  d1,41(a3)           ; Kontrollbyte schreiben
A444          bra.s   FEA44A              ; ---> Schrittschleife
A446
A446 FEA446   bsr     FEA4BE              ; ---> Kopfschritt
A44A FEA44A   dbra    d2,FEA446           ; wiederholen bis d2 = -1: ->
A44E          bsr     FEAE42              ; ---> Laufwerk freigeben
A452          move.l  30(a3),d0           ; d0 := tdu_SettleDelay
A456          bsr     FEA4F0              ; ---> d0 * 2 us warten
A45A FEA45A   moveq   #0,d0               ; d0 := 0 = ok-Flag
A45C FEA45C   movem.l (a7)+,d2-d3         ; Register wiederherstellen
A460          rts
A462
A462 ;--------------------------------   Motor schalten
A462
A462 FEA462   move.l  d2,-(a7)            ; d2 retten
A464          tst.l   d0                  ; d0 = 0?
```

```
A466          seq     d1                ; ja: d1 := $FF, nein: d1 := Ø
A468          andi.b  #8Ø,d1            ; DSKMOTOR-Bit isolieren
A46C          move.b  41(a3),d2         ; d2 := Kontrollbyte
A47Ø          andi.b  #8Ø,d2            ; DSKMOTOR-Bit isolieren
A474          cmp.b   d1,d2             ; Beide gleich?
A476          beq     FEA4B2            ; ja: Fertig ->
A47A          bclr.b  #7,41(a3)         ; DSKMOTOR-Bit löschen
A48Ø          or.b    d1,41(a3)         ; und neu einodern
A484          bsr     FEADDC            ; ---> Laufwerk anfordern
A488          moveq   #FF,d1            ; Alle Bits setzen
A48A          eor.b   d2,d1             ; DSKMOTOR-Bit := Ø, wenn es 1 war
A48C          move.b  d1,BFD1ØØ         ; Laufwerke deselektieren
A492          move.b  41(a3),BFD1ØØ     ; Kontrollbyte schreiben
A49A          bsr     FEAE42            ; ---> Laufwerk freigeben
A49E          btst.b  #7,41(a3)         ; Motor eingeschaltet?
A4A4          bne     FEA4B2            ; nein: ->
A4A8          move.l  #Ø7A12Ø,dØ        ; dØ := 5ØØØØØ
A4AE          bsr     FEA4FØ            ; ---> dØ * 2 us warten
A4B2 FEA4B2   moveq   #Ø,dØ             ; dØ := Ø
A4B4          btst.l  #7,d2             ; DSKMOTOR-Bit testen
A4B8          seq     dØ                ; dØ := Ø (aus) oder -1 (ein)
A4BA          move.l  (a7)+,d2          ; d2 wiederherstellen
A4BC          rts
A4BE
A4BE ;--------------------------------   Kopfschritt
A4BE
A4BE FEA4BE   move.b  41(a3),dØ         ; dØ := Kontrollbyte
A4C2          lea     BFD1ØØ,a1         ; a1 -> ciab.prb
A4C8          move.b  dØ,d1             ; d1 := dØ = Kontrollbyte
A4CA          bclr.l  #Ø,dØ             ; DSKSTEP-Bit := Ø
A4CE          move.b  dØ,(a1)           ; Kontrollbyte schreiben
A4DØ          nop                       ; ca. 1 us warten
A4D2          nop
A4D4          move.b  d1,(a1)           ; DSKSTEP-Bit := 1
A4D6          bsr     FEAE42            ; ---> Laufwerk freigeben
A4DA          move.l  2C(a3),dØ         ; dØ := tdu_StepDelay
A4DE          bsr     FEA4FØ            ; ---> dØ * 2 us warten
A4E2          bsr     FEADDC            ; ---> Laufwerk anfordern
```

```
A4E6          move.b  41(a3),BFD100       ; Kontrollbyte schreiben
A4EE          rts
A4F0
A4F0 ;--------------------------------   d0 * 2 us warten
A4F0
A4F0 FEA4F0   cmpi.l  #12C,d0             ; d0 < 300?
A4F6          blt     FEA51C              ; ja: d0 mal 2 us warten ->
A4FA          movem.l a0-a1,-(a7)         ; Register retten
A4FE          lea     5E(a3),a1           ; a1 -> IORequest unit_MicroHz
A502          clr.l   20(a1)              ; tr_Secs := 0
A506          move.l  d0,24(a1)           ; tr_Micro := d0
A50A          move.l  a6,-(a7)            ; a6 retten
A50C          movea.l 34(a6),a6           ; a6 := SysBase
A510          jsr     -1C8(a6)            ; ---> DoIO
A514          movea.l (a7)+,a6            ; a6 wiederherstellen
A516          movem.l (a7)+,a0-a1         ; Register wiederherstellen
A51A          bra.s   FEA522              ; ---> rts
A51C
A51C FEA51C   tst.l   (a3)                ; 4 Zyklen
A51E          dbra    d0,FEA51C           ; + 10 Zyklen = 2 us, d0 mal ->
A522 FEA522   rts
A524
A524 ;------ d0 Bytes von Disk einlesen und ab Adresse a0 speichern
A524
A524 FEA524   lea     84(pc),a1           ; a1 := FEA5AA -> DMA-Startroutine
A528 FEA528   movem.l d2/a2/a4,-(a7)      ; Register retten
A52C          movea.l a0,a2               ; a2 := a0 -> Datenbereich
A52E          move.l  d0,d2               ; d2 := d0 = Zahl der Bytes
A530          movea.l a1,a4               ; a4 -> DMA-Startroutine
A532          bsr     FEADDC              ; ---> Laufwerk anfordern
A536          move.b  41(a3),BFD100       ; Disk selektieren
A53E          move.w  #4000,DFF024        ; DSKLEN: DMA abschalten
A546          move.l  #3E8,d0             ; d0 := 1000
A54C          bsr.s   FEA4F0              ; ---> d0 * 2 us warten
A54E          lea     DFF000,a1           ; a1 := ChipBase
A554          move.l  a2,20(a1)           ; DSKPT := a2
A558          move.w  #1002,9C(a1)        ; INTREQ: CLR DSKSYN, DSKBLK
A55E          move.w  #8002,9A(a1)        ; INTENA: SET DSKBLK
A564          btst.b  #2,BFE001           ; DSKCHANGE gesetzt?
A56C          bne.s   FEA572              ; nein: ->
```

```
A56E FEA56E  moveq   #1D,d2              ; d2 := tderr_DiskChanged
A57Ø         bra.s   FEA59E              ; ---> Abschluß
A572
A572 FEA572  lsr.w   #1,d2               ; d2 := Zahl der Worte
A574         ori.w   #8ØØØ,d2            ; DMAEN-Bit setzen, WRITE-Bit löschen
A578         move.l  d2,dØ               ; dØ := d2
A57A         jsr     (a4)                ; ---> DSKLEN schreiben, DMA starten
A57C         bsr     FEA7ØA              ; ---> Warten auf RÜckmeldung
A58Ø         lea     DFFØØØ,a1           ; a1 := ChipBase
A586         move.w  #2,9A(a1)           ; INTENA: CLR DSKBLK
A58C         move.w  #4ØØØ,24(a1)        ; DSKLEN: DMA abschalten
A592         btst.b  #2,BFEØØ1           ; DSKCHANGE gesetzt?
A59A         beq.s   FEA56E              ; ja: Fehler ->
A59C         moveq   #Ø,d2               ; d2 := Ø
A59E FEA59E  bsr     FEAE42              ; ---> Laufwerk freigeben
A5A2         move.l  d2,dØ               ; dØ := d2 = Zahl der Bytes oder Fehler
A5A4         movem.l (a7)+,d2/a2/a4      ; Register wiederherstellen
A5A8         rts
A5AA
A5AA ;------ DSKLEN schreiben, DMA starten
A5AA
A5AA FEA5AA  move.w  dØ,24(a1)           ; DSKLEN mit dØ belegen
A5AE         move.w  dØ,24(a1)           ; DMA starten
A5B2         rts
A5B4
A5B4 ;------ dØ Bytes ab Adresse aØ auf Disk schreiben
A5B4
A5B4 FEA5B4  lea     -C(pc),a1           ; a1 := FEA5AA
A5B8 FEA5B8  movem.l d2/a2/a4,-(a7)      ; Register retten
A5BC         movea.l aØ,a2               ; a2 := aØ -> Datenbereich
A5BE         move.l  dØ,d2               ; d2 := dØ = Zahl der Bytes
A5CØ         movea.l a1,a4               ; a4 := a1 -> DMA-Startroutine
A5C2         bsr     FEADDC              ; ---> Laufwerk anfordern
A5C6         move.b  41(a3),BFD1ØØ       ; Disk selektieren
A5CE         move.w  #4ØØØ,DFFØ24        ; DSKLEN: DMA abschalten
A5D6         move.l  #3E8,dØ             ; dØ := 1ØØØ
A5DC         bsr     FEA4FØ              ; ---> dØ * 2 us warten
A5EØ         lea     DFFØØØ,a1           ; a1 := ChipBase
A5E6         move.l  a2,2Ø(a1)           ; DSKPT := a2
A5EA         move.w  #1ØØ2,9C(a1)        ; INTREQ: CLR DSKSYN, DSKBLK
```

```
A5F0          move.w  #8002,9A(a1)        ; INTENA: SET DSKBLK
A5F6          btst.b  #2,BFE001           ; DSKCHANGE gesetzt?
A5FE          bne.s   FEA606              ; nein: ->
A600 FEA600   moveq   #1D,d2              ; d2 := tderr_DiskChanged
A602          bra     FEA6E2              ; ---> Abschluß
A606
A606 FEA606   btst.b  #3,BFE001           ; DSKPROT gesetzt?
A60E          beq     FEA6EE              ; ja: Fehler ->
A612          movea.l 4,a0                ; a0 := SysBase
A616          move.w  #4000,DFF09A        ; Interrupts sperren
A61E          addq.b  #1,126(a0)          ; IDNestCnt erhöhen
A622          btst.b  #4,22(a6)           ; Reset-Flag gesetzt?
A628          beq.s   FEA642              ; nein: ->
A62A          moveq   #23,d2              ; d2 := tderr_PostReset
A62C          movea.l 4,a0                ; a0 := SysBase
A630          subq.b  #1,126(a0)          ; IDNestCnt erniedrigen
A634          bge.s   FEA63E              ; Interrupts noch gesperrt: ->
A636          move.w  #C000,DFF09A        ; Interrupts freigeben
A63E FEA63E   bra     FEA6E2              ; ---> Abschluß
A642
A642 FEA642   bset.b  #5,22(a6)           ; Schreibflag setzen
A648          movea.l 4,a0                ; a0 := SysBase
A64C          subq.b  #1,126(a0)          ; IDNestCnt erniedrigen
A650          bge.s   FEA65A              ; Interrupts noch gesperrt: ->
A652          move.w  #C000,DFF09A        ; Interrupts freigeben
A65A FEA65A   move.w  #6000,9E(a1)        ; ADKCON: CLR WORDSYNC, MSBSYNC
A660          move.w  4C(a3),d0           ; d0 := Spurnummer
A664          move.w  #8000,d1            ; PreComp = 0
A668          cmp.w   26(a3),d0           ; Spurnummer <= Comp01Track?
A66C          bls.s   FEA686              ; ja: ->
A66E          move.w  #A000,d1            ; PreComp = 140 ns
A672          cmp.w   28(a3),d0           ; Spurnummer <= Comp10Track?
A676          bls.s   FEA686              ; ja: ->
A678          move.w  #C000,d1            ; PreComp = 280 ns
A67C          cmp.w   2A(a3),d0           ; Spurnummer <= Comp11Track?
A680          bls.s   FEA686              ; ja: ->
A682          move.w  #E000,d1            ; PreComp = 560 ns
A686 FEA686   move.w  d1,9E(a1)           ; ADKCON: PreComp-Bits setzen
A68A          lsr.w   #1,d2               ; d2 := Zahl der Worte
A68C          ori.w   #C000,d2            ; DMAEN, WRITE-Bit setzen
```

```
A69Ø          move.l  d2,dØ               ; dØ := d2
A692          jsr     (a4)                ; ---> DSKLEN schreiben, DMA starten
A694          bsr     FEA7ØA              ; ---> Warten auf RÜckmeldung
A698          lea     DFFØØØ,a1           ; a1 := ChipBase
A69E          move.w  #2,9A(a1)           ; INTENA: CLR DSKBLK
A6A4          move.l  #7DØ,dØ             ; dØ := 2ØØØ
A6AA          bsr     FEA4FØ              ; ---> dØ * 2 us warten
A6AE          move.w  #4ØØØ,24(a1)        ; DSKLEN: DMA abschalten
A6B4          move.b  22(a6),dØ           ; dØ := Flags
A6B8          bclr.b  #5,22(a6)           ; Schreibflag löschen
A6BE          btst.l  #4,dØ               ; Reset-Flag gesetzt?
A6C2          beq.s   FEA6D4              ; nein: ->
A6C4          lea     78(a6),a1           ; a1 -> IORequest ResetHandler
A6C8          move.l  a6,-(a7)            ; a6 retten
A6CA          movea.l 14(a1),a6           ; a6 -> keyboard.device
A6CE          jsr     -1E(a6)             ; ---> BeginIO
A6D2          movea.l (a7)+,a6            ; a6 wiederherstellen
A6D4 FEA6D4   btst.b  #2,BFEØØ1           ; DSKCHANGE gesetzt?
A6DC          beq     FEA6ØØ              ; ja: Fehler ->
A6EØ          moveq   #Ø,d2               ; d2 := Ø = ok-Flag
A6E2 FEA6E2   bsr     FEAE42              ; ---> Laufwerk freigeben
A6E6          move.l  d2,dØ               ; dØ := d2 = Fehlerkode
A6E8          movem.l (a7)+,d2/a2/a4      ; Register wiederherstellen
A6EC          rts
A6EE
A6EE FEA6EE   moveq   #1C,d2              ; d2 := tderr_WriteProt
A6FØ          bra.s   FEA6E2              ; ---> Abschluß
A6F2
A6F2 ;--------------------------------   ReplyMessage senden
A6F2
A6F2 FEA6F2   lea     AE(a1),aØ           ; aØ -> ReplyPort
A6F6          lea     5E(a1),a1           ; a1 -> IORequest unit_MicroHz
A6FA          move.l  a6,-(a7)            ; a6 retten
A6FC          movea.l 4,a6                ; a6 := SysBase
A7Ø2          jsr     -16E(a6)            ; ---> PutMsg
A7Ø6          movea.l (a7)+,a6            ; a6 wiederherstellen
A7Ø8          rts
A7ØA
```

```
A70A ;--------------------------------   Auf Rückmeldung am ReplyPort warten
A70A
A70A FEA70A  move.l  #400,d0             ; d0 := Wecksignal Bit 10
A710         move.l  a6,-(a7)            ; a6 retten
A712         movea.l 34(a6),a6           ; a6 := SysBase
A716         jsr     -13E(a6)            ; ---> Wait
A71A         movea.l (a7)+,a6            ; a6 wiederherstellen
A71C         lea     AE(a3),a0           ; a0 -> ReplyPort
A720         move.l  a6,-(a7)            ; a6 retten
A722         movea.l 34(a6),a6           ; a6 := SysBase
A726         jsr     -174(a6)            ; ---> GetMsg
A72A         movea.l (a7)+,a6            ; a6 wiederherstellen
A72C         tst.l   d0                  ; Message vorhanden?
A72E         beq.s   FEA70A              ; nein: Wiederholen ->
A730         rts
A732
A732         DC.W    0
A734
A734 ;--------------------------------   cmd_Read, cmd_Write
A734
A734 FEA734  movem.l a2-a4,-(a7)         ; Register retten
A738         movea.l 18(a1),a3           ; a3 := io_Unit
A73C         movea.l a1,a2               ; a2 := a1 -> IORequest
A73E         move.l  a2,44(a3)           ; in trackdisk.unit eintragen
A742         move.l  #0,20(a2)           ; io_Actual := 0
A74A         move.l  28(a2),56(a3)       ; io_Data in trackdisk.unit eintragen
A750         move.l  2C(a2),d0           ; d0 := io_Offset
A754         bsr     FEA182              ; ---> Spur/Sektor aus Offset ermitteln
A758         tst.l   d0                  ; zulässiges Ergebnis?
A75A         bmi     FEA92A              ; nein: Fehler ->
A75E         move.w  d0,4A(a3)           ; Spurnummer eintragen
A762         move.b  d1,49(a3)           ; Sektornummer eintragen
A766         move.l  2C(a2),d0           ; d0 := io_Offset
A76A         add.l   24(a2),d0           ; + io_Length
A76E         bsr     FEA182              ; ---> Spur/Sektor aus d0 ermitteln
A772         tst.l   d0                  ; zulässiges Ergebnis?
A774         bmi     FEA92A              ; nein: Fehler ->
A778         btst.b  #2,40(a3)           ; ExtCmd-Flag gesetzt?
A77E         beq.s   FEA78E              ; nein: ->
A780         move.l  34(a2),5A(a3)       ; iotd_SecLabel in trackdisk.unit eintr.
```

```
A786          beq.s   FEA78E            ; = 0: ->
A788          bset.b  #0,40(a3)        ; SecLabel-Flag setzen
A78E FEA78E   btst.b  #1,40(a3)        ; NoDisk-Flag gesetzt?
A794          beq.s   FEA7A0           ; nein: ->
A796          move.b  #1D,1F(a2)       ; io_Error := tderr_DiskChanged
A79C          bra     FEA920           ; ---> Abschluß
A7A0
A7A0 FEA7A0   move.w  4A(a3),d0        ; d0 := Spurnummer
A7A4          bsr     FEA93C           ; ---> Wenn Spur im Puffer, a0 -> Puffer
A7A8          move.l  a0,4E(a3)        ; a0 eintragen
A7AC          bne.s   FEA804           ; a0 nicht 0: Spur im Puffer ->
A7AE          bsr     FEA952           ; ---> a0 -> Spurpuffer
A7B2          movea.l a0,a2            ; a2 := a0 -> Spurpuffer
A7B4          move.l  a2,4E(a3)        ; Anfang des Datenbereichs := a2
A7B8 FEA7B8   btst.b  #0,2(a2)         ; Änderungsflag gesetzt?
A7BE          beq.s   FEA7D4           ; nein: ->
A7C0          bsr     FEA958           ; ---> Spur im Puffer auf Disk schreiben
A7C4          tst.l   d0               ; ok?
A7C6          beq.s   FEA7D4           ; ja: ->
A7C8          movea.l 44(a3),a1        ; a1 -> IORequest
A7CC          move.b  d0,1F(a1)        ; io_Error := d0
A7D0          bra     FEA920           ; ---> Abschluß
A7D4
A7D4 FEA7D4   move.w  4A(a3),0(a2)     ; Spurnummer an Pufferanfang setzen
A7DA          bclr.b  #0,2(a2)         ; Änderungsflag löschen
A7E0          clr.b   42(a3)           ; Sektorzähler rücksetzen
A7E4          bsr     FEA99E           ; ---> Spur in Puffer einlesen
A7E8          move.b  3(a2),d0         ; d0 := Sektornummer
A7EC          cmpi.b  #B,d0            ; Sektornummer < 11?
A7F0          bcs.s   FEA804           ; ja: ->
A7F2          move.w  #-1,0(a2)        ; Spurnummer ungültig machen
A7F8          movea.l 44(a3),a1        ; a1 -> IORequest
A7FC          move.b  d0,1F(a1)        ; io_Error := d0
A800          bra     FEA920           ; ---> Abschluß
A804
A804 FEA804   movea.l 44(a3),a2        ; a2 -> IORequest
A808          move.w  1C(a2),d0        ; d0 := io_Command
A80C          movea.l 4E(a3),a0        ; a0 -> Spurpuffer
```

```
A81Ø          cmpi.b  #3,dØ               ; dØ = cmd_Write?
A814          bne     FEA89Ø              ; nein: ->
A818
A818 ;------ Write
A818
A818          bset.b  #Ø,2(aØ)            ; Änderungsflag setzen
A81E          moveq   #Ø,dØ               ; dØ := Ø
A82Ø          move.b  49(a3),dØ           ; dØ := Sektornummer
A824          sub.b   3(aØ),dØ            ; - Nummer des 1. Sektors im Puffer
A828          bpl.s   FEA82E              ; nicht negativ: ->
A82A          addi.b  #B,dØ               ; 11 addieren
A82E FEA82E   mulu    #44Ø,dØ             ; dØ := dØ * 1Ø88
A832          lea     68Ø(aØ),a4          ; a4 -> Spurpuffer + 1664 (Lücke)
A836          adda.l  dØ,a4               ; + dØ: a4 -> gesuchter Sektor
A838          btst.b  #Ø,4Ø(a3)           ; SecLabel-Flag gesetzt?
A83E          beq.s   FEA866              ; nein: ->
A84Ø          movea.l 5A(a3),aØ           ; aØ := iotd_SecLabel
A844          move.l  #1Ø,dØ              ; dØ := 16 Bytes
A84A          lea     1Ø(a4),a1           ; a1 := a4 + 16 -> SecLabel im Header
A84E          bsr     FEAB4A              ; ---> dØ Bytes kodieren und speichern
A852          lea     8(a4),aØ            ; aØ := a4 + 8 -> Header
A856          move.w  #28,d1              ; d1 := 4Ø Bytes
A85A          bsr     FEADA4              ; ---> Prüfsumme in dØ bilden
A85E          lea     3Ø(a4),aØ           ; aØ := a4 + 48 -> Header-Prüfsumme
A862          bsr     FEAD46              ; ---> dØ kodieren und speichern
A866 FEA866   movea.l 56(a3),aØ           ; aØ := io_Data
A86A          move.l  #2ØØ,dØ             ; dØ := 512 Bytes
A87Ø          lea     4Ø(a4),a1           ; a1 := a4 + 64 -> Block-Anfang
A874          bsr     FEAB4A              ; ---> dØ Bytes kodieren und speichern
A878          lea     4Ø(a4),aØ           ; aØ := a4 + 64 -> Block-Anfang
A87C          move.w  #4ØØ,d1             ; d1 := 1Ø24 Bytes MFM-Kode
A88Ø          bsr     FEADA4              ; ---> Prüfsumme in dØ bilden
A884          lea     38(a4),aØ           ; aØ := a4 + 56 -> Block-Prüfsumme
A888          bsr     FEAD46              ; ---> dØ kodieren und speichern
A88C          bra     FEA8D6              ; --->
A89Ø
A89Ø ;------ Read
A89Ø
A89Ø FEA89Ø   moveq   #Ø,dØ               ; dØ := Ø
A892          move.b  49(a3),dØ           ; dØ := Sektornummer
```

```
A896          sub.b   3(aØ),dØ          ; - Nummer des 1. Sektors im Puffer
A89A          bpl.s   FEA8AØ            ; nicht negativ: ->
A89C          addi.b  #B,dØ             ; 11 addieren
A8AØ FEA8AØ   mulu    #44Ø,dØ           ; dØ := dØ * 1Ø88
A8A4          lea     68Ø(aØ),a4        ; a4 -> 1. Sektor im Puffer
A8A8          adda.l  dØ,a4             ; + dØ -> gesuchter Sektor
A8AA          btst.b  #Ø,4Ø(a3)         ; SecLabel-Flag gesetzt?
A8BØ          beq.s   FEA8C4            ; nein: ->
A8B2          lea     1Ø(a4),a1         ; a1 := a4 + 16 -> SecLabel im Sektor
A8B6          movea.l 5A(a3),aØ         ; aØ := iotd_SecLabel
A8BA          move.l  #1Ø,dØ            ; dØ := 16
A8CØ          bsr     FEACB2            ; ---> dØ Worte aus Puffer dekodieren
A8C4 FEA8C4   lea     4Ø(a4),a1         ; a1 := a4 + 64 -> Block-Anfang
A8C8          movea.l 56(a3),aØ         ; aØ := io_Data
A8CC          move.l  #2ØØ,dØ           ; dØ := 512 Worte MFM-Kode
A8D2          bsr     FEACB2            ; ---> dØ Worte aus Puffer dekodieren
A8D6
A8D6 ;------ Read & Write
A8D6
A8D6 FEA8D6   move.l  #2ØØ,d1           ; d1 := 512 Bytes
A8DC          add.l   d1,56(a3)         ; zum Datenzeiger addieren
A8EØ          move.l  2Ø(a2),dØ         ; dØ := io_Actual
A8E4          add.l   d1,dØ             ; Blocklänge addieren
A8E6          move.l  dØ,2Ø(a2)         ; io_Actual := Ergebnis
A8EA          btst.b  #Ø,4Ø(a3)         ; SecLabel-Flag gesetzt?
A8FØ          beq.s   FEA8FA            ; nein: ->
A8F2          addi.l  #1Ø,5A(a3)        ; 16 zum SecLabel-Zeiger addieren
A8FA FEA8FA   cmp.l   24(a2),dØ         ; dØ < io_Length?
A8FE          bcc.s   FEA92Ø            ; nein: ->
A9ØØ          movea.l 4E(a3),a2         ; a2 -> Spurpuffer
A9Ø4          addq.b  #1,49(a3)         ; Sektornummer erhöhen
A9Ø8          cmpi.b  #B,49(a3)         ; Sektornummer < 11?
A9ØE          blt     FEA8Ø4            ; ja: ->
A912          move.b  #Ø,49(a3)         ; Sektornummer := Ø
A918          addq.w  #1,4A(a3)         ; Spurnummer erhöhen
A91C          bra     FEA7B8            ; ---> Loop
A92Ø
A92Ø FEA92Ø   movea.l 44(a3),a1         ; a1 -> IORequest
A924          bsr     FEA1BØ            ; ---> IORequest beantworten
A928          bra.s   FEA936            ; ---> Abschluß
```

```
A92A
A92A FEA92A  movea.l 44(a3),a1           ; a1 -> IORequest
A92E         move.b  #FC,1F(a1)          ; io_Error := ioerr_BadLength
A934         bra.s   FEA92Ø              ; ---> Abschluß
A936
A936 FEA936  movem.l (a7)+,a2-a4         ; Register wiederherstellen
A93A         rts
A93C
A93C ;------ Prüfen ob Spur Nummer dØ im Puffer
A93C
A93C FEA93C  move.l  52(a3),d1           ; d1 := Pufferadresse
A94Ø         tst.l   d1                  ; = Ø?
A942         beq.s   FEA94C              ; ja: ->
A944         movea.l d1,aØ               ; aØ -> Spurpuffer
A946         cmp.w   Ø(aØ),dØ            ; dØ = Nummer der Spur im Puffer?
A94A         beq.s   FEA95Ø              ; ja: ->
A94C FEA94C  moveq   #Ø,dØ               ; dØ := Ø
A94E         movea.l dØ,aØ               ; aØ := Ø
A95Ø FEA95Ø  rts
A952
A952 ;------ Adresse des Spurpuffers nach aØ
A952
A952 FEA952  movea.l 52(a3),aØ           ; aØ -> Spurpuffer
A956         rts
A958
A958 ;--------------------------------   Spur auf Disk schreiben
A958
A958 FEA958  movem.l d2/a2,-(a7)         ; Register retten
A95C         movea.l 4E(a3),a2           ; a2 -> Spurpuffer
A96Ø         moveq   #1,dØ               ; dØ := 1
A962         bsr     FEA462              ; ---> Motor einschalten
A966         moveq   #Ø,dØ               ; dØ := Ø
A968         move.w  Ø(a2),dØ            ; dØ := Spurnummer aus Puffer
A96C         bsr     FEA3DA              ; ---> Kopf auf Spur dØ setzen
A97Ø         lea     4(a2),aØ            ; aØ -> Spurdaten im Puffer
A974         move.w  #353E,dØ            ; dØ := 13630 Bytes Spurlänge
A978         bsr     FEA5B4              ; ---> dØ Bytes ab Adresse aØ schreiben
A97C         movem.l (a7)+,d2/a2 ; Register wiederherstellen
A98Ø         rts
A982
```

```
A982 ;--------------------------------  Fehlertext (nicht benutzt)
A982
A982 FEA982  DC.B    'read error %ld, track %d',0D,0A,0,0
A99E
A99E ;--------------------------------  Spur von Disk lesen
A99E
A99E FEA99E  movem.l a2,-(a7)          ; a2 retten
A9A2         movea.l 4E(a3),a2         ; a2 -> Spurpuffer
A9A6         moveq   #1,d0             ; d0 := 1
A9A8         bsr     FEA462            ; ---> Motor einschalten
A9AC FEA9AC  moveq   #0,d0             ; d0 := 0
A9AE         move.w  4A(a3),d0         ; d0 := Spurnummer
A9B2         bsr     FEA3DA            ; ---> Kopf auf Spur d0 setzen
A9B6 FEA9B6  lea     684(a2),a0        ; a0 -> Spurpuffer + 1668 (Lücke)
A9BA         move.w  #397C,d0          ; d0 := 14716
A9BE         bsr     FEA524            ; ---> d0 Bytes ab Adresse a0 einlesen
A9C2         tst.l   d0                ; Fehler?
A9C4         beq.s   FEA9CE            ; nein: ->
A9C6         move.b  d0,3(a2)          ; Fehlerkode eintragen
A9CA         bra     FEA9F8            ; ---> Abschluß
A9CE
A9CE FEA9CE  bsr     FEAFE2            ; ---> Spur normalisieren und prüfen
A9D2         move.b  d0,3(a2)          ; Nummer des 1. Sektors eintragen
A9D6         cmpi.b  #B,d0             ; Sektornummer < 11?
A9DA         bcs.s   FEA9F8            ; ja: Fertig ->
A9DC         addq.b  #1,42(a3)         ; Wiederholungszähler erhöhen
A9E0         move.b  42(a3),d0         ; d0 := Wiederholungszählerstand
A9E4         cmp.b   34(a3),d0         ; > tdu_RetryCnt?
A9E8         bgt.s   FEA9F8            ; ja: Abschluß ->
A9EA FEA9EA  andi.b  #3,d0             ; d0 := Wiederholungszähler mod 4
A9EE         bne.s   FEA9B6            ; nicht 0: Lesen wiederholen ->
A9F0         move.w  #-1,4C(a3)        ; Spurnummer ungültig machen
A9F6         bra.s   FEA9AC            ; ---> Kopf neu setzen und wiederholen
A9F8
A9F8 FEA9F8  movem.l (a7)+,a2          ; a2 wiederherstellen
A9FC         rts
A9FE
```

```
A9FE ;--------------------------------   td_Motor
A9FE
A9FE FEA9FE  move.l  a1,-(a7)            ; a1 retten
AA00         move.l  24(a1),d0           ; d0 := Motorstatus (0 = aus)
AA04         bsr     FEA462              ; ---> Motor schalten
AA08         movea.l (a7)+,a1            ; a1 wiederherstellen
AA0A         move.l  d0,20(a1)           ; io_Actual := Motorstatus
AA0E         bsr     FEA1B0              ; ---> IORequest beantworten
AA12         rts
AA14
AA14 ;--------------------------------   td_Seek
AA14
AA14 FEAA14  move.l  a2,-(a7)            ; a2 retten
AA16         movea.l a1,a2               ; a2 := a1 -> IORequest
AA18         move.l  2C(a1),d0           ; d0 := io_Offset
AA1C         bsr     FEA182              ; ---> Spur/Sektor für Offset ermitteln
AA20         tst.l   d0                  ; zulässiges Ergebnis?
AA22         bmi.s   FEAA32              ; nein: Fehler ->
AA24         bsr     FEA3DA              ; ---> Kopf auf Spur d0 setzen
AA28 FEAA28  movea.l a2,a1               ; a1 := a2 -> IORequest
AA2A         bsr     FEA1B0              ; ---> IORequest beantworten
AA2E         movea.l (a7)+,a2            ; a2 wiederherstellen
AA30         rts
AA32
AA32 FEAA32  move.b  #FC,1F(a2)          ; io_Error := ioerr_BadLength
AA38         bra.s   FEAA28              ; ---> Abschluß
AA3A
AA3A FEAA3A  move.b  #1D,1F(a1)          ; io_Error := tderr_DiskChanged
AA40         bra     FEAA88              ; ---> Abschluß
AA44
AA44 ;--------------------------------   td_ProtStatus
AA44
AA44 FEAA44  movem.l d2/a2-a3,-(a7)      ; Register retten
AA48         movea.l a1,a2               ; a2 := a1 -> IORequest
AA4A         movea.l 18(a2),a3           ; a3 := io_Unit
AA4E         btst.b  #1,40(a3)           ; NoDisk-Flag gesetzt?
AA54         bne.s   FEAA3A              ; ja: Fehler ->
AA56         moveq   #1,d0               ; d0 := 1
AA58         bsr     FEA462              ; ---> Motor einschalten
AA5C         bsr     FEADDC              ; ---> Laufwerk anfordern
```

```
AA6Ø          move.b  41(a3),BFD1ØØ       ; Kontrollbyte schreiben
AA68          nop                         ; ca. 1 us warten
AA6A          nop
AA6C          move.b  BFEØØ1,d2           ; d2 := Diskstatus
AA72          bsr     FEAE42              ; ---> Laufwerk freigeben
AA76          moveq   #Ø,dØ               ; dØ := Ø
AA78          btst.l  #3,d2               ; WriteProtect-Flag gesetzt?
AA7C          seq     dØ                  ; ja: dØ := -1, nein: dØ := Ø
AA7E          move.l  dØ,2Ø(a2)           ; io_Actual := dØ
AA82          moveq   #Ø,dØ               ; dØ := Ø
AA84          bsr     FEA462              ; ---> Motor ausschalten
AA88 FEAA88   movea.l a2,a1               ; a1 := a2 -> IORequest
AA8A          bsr     FEA1BØ              ; ---> IORequest beantworten
AA8E          movem.l (a7)+,d2/a2-a3      ; Register retten
AA92          rts
AA94
AA94 ;--------------------------------   cmd_Clear
AA94
AA94 FEAA94   movea.l 52(a3),aØ           ; aØ -> Spurpuffer
AA98          move.w  #-1,Ø(aØ)           ; Spurnummer im Puffer ungültig machen
AA9E          bclr.b  #Ø,2(aØ)            ; Änderungsflag löschen
AAA4          bsr     FEA1BØ              ; ---> IORequest beantworten
AAA8          rts
AAAA
AAAA ;--------------------------------   cmd_Update
AAAA
AAAA FEAAAA   move.l  a2,-(a7)            ; a2 retten
AAAC          movea.l a1,a2               ; a2 := a1 -> IORequest
AAAE          movea.l 52(a3),aØ           ; aØ -> Spurpuffer
AAB2          btst.b  #Ø,2(aØ)            ; Änderungsflag gesetzt?
AAB8          beq.s   FEAAD4              ; nein: Fertig ->
AABA          bsr     FEA958              ; ---> Spur auf Disk schreiben
AABE          movea.l a2,a1               ; a1 := a2 -> IORequest
AACØ          tst.l   dØ                  ; Fehler?
AAC2          beq.s   FEAACA              ; nein: ->
AAC4          move.b  dØ,1F(a1)           ; io_Error := Fehlerkode
AAC8          bra.s   FEAAD4              ; ---> Abschluß
AACA
AACA FEAACA   movea.l 52(a3),aØ           ; aØ -> Spurpuffer
AACE          bclr.b  #Ø,2(aØ)            ; Änderungsflag löschen
```

```
AAD4 FEAAD4  bsr     FEA1B0              ; ---> IORequest beantworten
AAD8         movea.l (a7)+,a2            ; a2 wiederherstellen
AADA         rts
AADC
AADC ;-------------------------------    Sektordaten im Puffer erzeugen
AADC
AADC FEAADC  movem.l d2/a2/a4,-(a7)      ; Register retten
AAE0         movea.l a1,a4               ; a4 := a1 -> Puffer
AAE2         movea.l a0,a2               ; a2 := a0 = io_Data
AAE4         move.l  d0,d2               ; d2 := d0 = Header-Langwort
AAE6         moveq   #0,d0               ; d0 := 0
AAE8         lea     0(a4),a0            ; a0 -> Puffer
AAEC         bsr     FEAD46              ; ---> d0 kodieren und speichern
AAF0         move.l  #44894489,4(a4)     ; Synchronworte in Datenbereich
AAF8         move.l  d2,d0               ; d0 := d2 = Header-Langwort
AAFA         lea     8(a4),a0            ; a0 -> Header im Puffer
AAFE         bsr     FEAD46              ; ---> d0 kodieren und speichern
AB02         moveq   #3,d2               ; d2 := 3 = Zähler-Anfangswert
AB04 FEAB04  moveq   #0,d0               ; d0 := 0
AB06         bsr     FEAD46              ; ---> d0 kodieren und speichern
AB0A         dbra    d2,FEAB04           ; dreimal wiederholen ->
AB0E         lea     8(a4),a0            ; a0 -> Header im Puffer
AB12         moveq   #28,d1              ; d1 := 40 Bytes
AB14         bsr     FEADA4              ; ---> Prüfsumme in d0 berechnen
AB18         lea     30(a4),a0           ; d0 -> Header-Prüfsumme im Puffer
AB1C         bsr     FEAD46              ; ---> d0 kodieren und speichern
AB20         move.l  #200,d0             ; d0 := 512 = Zahl der Datenbytes
AB26         movea.l a2,a0               ; a0 := a2 = io_Data
AB28         lea     40(a4),a1           ; a1 -> Sektordaten im Puffer
AB2C         bsr     FEAB4A              ; ---> Sektordaten kodieren u. speichern
AB30         lea     40(a4),a0           ; a0 -> Sektordaten im Puffer
AB34         move.w  #400,d1             ; d1 := 1024 = Zahl der MFM-Bytes
AB38         bsr     FEADA4              ; ---> Prüfsumme in d0 berechnen
AB3C         lea     38(a4),a0           ; a0 -> Block-Prüfsumme im Puffer
AB40         bsr     FEAD46              ; ---> d0 kodieren und speichern
AB44         movem.l (a7)+,d2/a2/a4      ; Register wiederherstellen
AB48         rts
AB4A
```

```
AB4A ;------------------------  dØ Bytes ab aØ kodieren und ab a1 speichern
AB4A
AB4A FEAB4A  link    a2,#-1E             ; Stackrahmen 3Ø Bytes für BltNode
AB4E         move.w  dØ,d1               ; d1 := dØ = Bytezahl
AB5Ø         lsl.w   #2,d1               ; BltSize h = Bytezahl/16
AB52         ori.w   #8,d1               ;         w = 8 Worte = 16 Bytes
AB56         move.w  d1,-A(a2)           ; BltSize in BltNode eintragen
AB5A         movem.l dØ/aØ-a1,-16(a2)    ; Bytezahl, Quelle, Ziel eintragen
AB6Ø         move.l  #FEAB9E,-1A(a2)     ; Blitter-Funktion eintragen
AB68         move.l  a3,-4(a2)           ; trackdisk.unit eintragen
AB6C         lea     -1E(a2),a1          ; a1 -> BltNode
AB7Ø         move.l  a6,-(a7)            ; a6 retten
AB72         movea.l 38(a6),a6           ; a6 := GfxBase
AB76         jsr     -114(a6)            ; ---> QBlit
AB7A         movea.l (a7)+,a6            ; a6 wiederherstellen
AB7C         bsr     FEA7ØA              ; ---> Auf Rückmeldung warten
AB8Ø         movem.l -16(a2),dØ/aØ-a1    ; Register wiederherstellen
AB86         move.l  dØ,d1               ; dØ := d1 = Bytezahl
AB88         movea.l a1,aØ               ; aØ := a1 -> ZielAnfang
AB8A         bsr     FEADBE              ; ---> Grenzkorrektur
AB8E         adda.l  d1,aØ               ; aØ -> ZielMitte
AB9Ø         bsr     FEADBE              ; ---> Grenzkorrektur
AB94         adda.l  d1,aØ               ; aØ -> ZielEnde
AB96         bsr     FEADBE              ; ---> Grenzkorrektur
AB9A         unlk    a2                  ; Stackrahmen freigeben
AB9C         rts
AB9E
AB9E ;------ Blitter-Funktionen
AB9E
AB9E FEAB9E  move.l  a5,-(a7)            ; a5 retten
ABAØ         movea.l a1,a5               ; a5 := a1 -> BltNode
ABA2         bsr     FEB2CC              ; ---> Masken und Modulo setzen
ABA6         movea.l a5,a1               ; a1 := a5 -> BltNode
ABA8         movem.l 8(a1),dØ-d1/a5      ; dØ := Bytes, d1 -> Quelle, a5 -> Ziel
ABAE         move.l  d1,4C(aØ)           ; BLTBPNT := d1 (Quelle B)
ABB2         move.l  d1,5Ø(aØ)           ; BLTAPNT := d1 (Quelle A)
ABB6         move.l  a5,54(aØ)           ; BLTDPNT := a5 (Ziel D)
ABBA         move.w  #1DB1,4Ø(aØ)        ; BLTCONØ := ASHØ+USEA,B,D+LF7,5,4,Ø
ABCØ         move.w  #Ø,42(aØ)           ; BLTCON1 := Ø
ABC6         move.w  14(a1),58(aØ)       ; BLTSIZE schreiben startet Blitter
```

```
ABCC          move.l  #FEABD8,4(a1)       ; Nächste Funktion in BltNode eintragen
ABD4          movea.l (a7)+,a5            ; a5 wiederherstellen
ABD6          rts
ABD8
ABD8 FEABD8   move.l  a5,-(a7)            ; a5 retten
ABDA          movem.l 8(a1),d0-d1/a5      ; d0 := Bytes, d1 -> Quelle, a5 -> Ziel
ABE0          move.l  a5,4C(a0)           ; BLTBPNT := a5 (Quelle B)
ABE4          move.l  d1,50(a0)           ; BLTAPNT := d1 (Quelle A)
ABE8          move.l  a5,54(a0)           ; BLTDPNT := a5 (Ziel D)
ABEC          move.w  #2D8C,40(a0)        ; BLTCON0 := ASH1+USEA,B,D+LF7,3,2
ABF2          move.w  14(a1),58(a0)       ; BLTSIZE schreiben startet Blitter
ABF8          move.l  #FEAC04,4(a1)       ; Nächste Funktion in BltNode eintragen
AC00          movea.l (a7)+,a5            ; a5 wiederherstellen
AC02          rts
AC04
AC04 FEAC04   move.l  a5,-(a7)            ; a5 retten
AC06          movem.l 8(a1),d0-d1/a5      ; d0 := Bytes, d1 -> Quelle, a5 -> Ziel
AC0C          add.l   d0,d1               ; d1 := Quelle Ende
AC0E          subq.l  #2,d1               ; d1 -> letztes Wort des Quellbereichs
AC10          adda.l  d0,a5               ; a5 -> Ziel Ende
AC12          adda.l  d0,a5
AC14          subq.l  #2,a5               ; a5 -> letztes Wort des Zielbereichs
AC16          move.l  d1,4C(a0)           ; BLTBPNT := d1 (Quelle B)
AC1A          move.l  d1,50(a0)           ; BLTAPNT := d1 (Quelle A)
AC1E          move.l  a5,54(a0)           ; BLTDPNT := a5 (Ziel D)
AC22          move.w  #DB1,40(a0)         ; BLTCON0 := USEA,B,D+LF7,5,4,0
AC28          move.w  #1002,42(a0)        ; BLTCON1 := BSH0+DESC
AC2E          move.w  14(a1),58(a0)       ; BLTSIZE schreiben startet Blitter
AC34          move.l  #FEAC40,4(a1)       ; Nächste Funktion in BltNode eintragen
AC3C          movea.l (a7)+,a5            ; a5 wiederherstellen
AC3E          rts
AC40
AC40 FEAC40   move.l  a5,-(a7)            ; a5 retten
AC42          movem.l 8(a1),d0-d1/a5      ; d0 := Bytes, d1 -> Quelle, a5 -> Ziel
AC48          adda.l  d0,a5               ; a5 -> Ziel Ende
AC4A          move.l  a5,4C(a0)           ; BLTBPNT := a5 (Quelle B)
AC4E          move.l  d1,50(a0)           ; BLTAPNT := d1 (Quelle A)
AC52          move.l  a5,54(a0)           ; BLTDPNT := a5 (Ziel D)
AC56          move.w  #1D8C,40(a0)        ; BLTCON0 := ASH0+USEA,B,D+LF7,3,2
AC5C          move.w  #0,42(a0)           ; BLTCON1 := 0
```

```
AC62          move.w  14(a1),58(aØ)       ; BLTSIZE schreiben startet Blitter
AC68          move.l  #FEAC74,4(a1)       ; Nächste Funktion in BltNode eintragen
AC7Ø          movea.l (a7)+,a5            ; a5 wiederherstellen
AC72          rts
AC74
AC74 FEAC74   moveq   #Ø,dØ               ; dØ := Ø
AC76          move.l  dØ,4(a1)            ; Keine weitere Funktion
AC7A          movea.l 1A(a1),a1           ; a1 -> trackdisk.unit
AC7E          bsr     FEA6F2              ; ---> Rückmeldung senden
AC82          moveq   #Ø,dØ               ; dØ := Ø
AC84          rts
AC86
AC86 ;--------------------------------   Routine, die nicht gerufen wird
AC86
AC86          movem.l d2/a2/a4,-(a7)      ; Register retten
AC8A          movea.l aØ,a2               ; a2 := aØ -> Bereich für Speicherung
AC8C          movea.l a1,a4               ; a4 := a1
AC8E          addq.l  #8,a4               ; + 8
AC9Ø          movea.l a4,aØ               ; aØ = Anfang für Dekodierung
AC92          bsr     FEAD8E              ; ---> 64 MFM-Bits in dØ dekodieren
AC96          move.l  dØ,d2               ; d2 := dØ
AC98          addq.l  #8,a4               ; Header-Prüfsumme übergehen
AC9A          addq.l  #8,a4               ; Block-Prüfsumme übergehen
AC9C          move.l  #2ØØ,dØ             ; dØ := 512 = Blocklänge in MFM-Worten
ACA2          movea.l a4,a1               ; a1 -> Blockanfang
ACA4          movea.l a2,aØ               ; aØ -> Datenbereich für Speicherung
ACA6          bsr     FEACB2              ; ---> dØ Worte aus Puffer dekodieren
ACAA          move.l  d2,dØ               ; dØ := d2
ACAC          movem.l (a7)+,d2/a2/a4      ; Register wiederherstellen
ACBØ          rts
ACB2
ACB2 ;-----------------------  dØ Worte ab a1 dekodieren und ab aØ speichern
ACB2
ACB2 FEACB2   link    a2,#-1E             ; Stackrahmen 3Ø Bytes für BltNode
ACB6          movem.l dØ/aØ-a1,-16(a2)    ; Bytezahl, Quelle, Ziel eintragen
ACBC          lsl.w   #2,dØ               ; BltSize h = Bytezahl/16
ACBE          ori.w   #8,dØ               ;         w = 8 Worte = 16 Bytes
ACC2          move.w  dØ,-A(a2)           ; BltSize in BltNode eintragen
ACC6          move.l  #FEACFØ,-1A(a2)     ; Blitter-Funktion eintragen
ACCE          move.l  a3,-4(a2)           ; trackdisk.unit eintragen
```

```
ACD2          lea      -1E(a2),a1          ; a1 -> BltNode
ACD6          move.l   a6,-(a7)            ; a6 retten
ACD8          movea.l  38(a6),a6           ; a6 := GfxBase
ACDC          jsr      -114(a6)            ; ---> QBlit
ACE0          movea.l  (a7)+,a6            ; a6 wiederherstellen
ACE2          bsr      FEA70A              ; ---> Auf Rückmeldung warten
ACE6          movem.l  -16(a2),d0/a0-a1    ; Register wiederherstellen
ACEC          unlk     a2                  ; Stackrahmen freigeben
ACEE          rts
ACF0
ACF0 ;------ Blitter-Funktionen
ACF0
ACF0 FEACF0   move.l   a5,-(a7)            ; a5 retten
ACF2          movea.l  a1,a5               ; a5 := a1 -> BltNode
ACF4          bsr      FEB2CC              ; ---> Masken und Modulo setzen
ACF8          movea.l  a5,a1               ; a1 := a5 -> BltNode
ACFA          movem.l  8(a1),d0-d1/a5      ; d0 := Bytes, d1 -> Ziel, a5 -> Quelle
AD00          adda.l   d0,a5               ; a5 -> QuelleMitte
AD02          subq.l   #1,a5               ; - 1
AD04          move.l   a5,50(a0)           ; BLTAPNT := a5 (Quelle A)
AD08          adda.l   d0,a5               ; a5 -> QuelleEnde
AD0A          move.l   a5,4C(a0)           ; BLTBPNT := a5 (Quelle B)
AD0E          add.l    d0,d1               ; d1 := d0 -> ZielEnde
AD10          subq.l   #1,d1               ; - 1
AD12          move.l   d1,54(a0)           ; BLTDPNT := d1 (Ziel D)
AD16          move.w   #1DD8,40(a0)        ; BLTCON0 := ASH0+USEA,B,D+LF7,6,4,3
AD1C          move.w   #2,42(a0)           ; BLTCON1 := DESC
AD22          move.w   14(a1),58(a0)       ; BLTSIZE schreiben startet Blitter
AD28          move.l   #FEAD34,4(a1)       ; Nächste Funktion in BltNode eintragen
AD30          movea.l  (a7)+,a5            ; a5 wiederherstellen
AD32          rts
AD34
AD34 FEAD34   moveq    #0,d0               ; d0 := 0
AD36          move.l   d0,4(a1)            ; Keine weitere Funktion
AD3A          movea.l  1A(a1),a1           ; a1 -> trackdisk.unit
AD3E          bsr      FEA6F2              ; ---> Rückmeldung senden
AD42          moveq    #0,d0               ; d0 := 0
AD44          rts
AD46
```

```
AD46 ;--------------------------------   dØ kodieren und ab aØ speichern
AD46
AD46 FEAD46  movem.l d2-d3,-(a7)         ; Register retten
AD4A         move.l  dØ,d3               ; d3 := dØ = Daten-Langwort
AD4C         lsr.l   #1,dØ               ; dØ um 1 Bit rechtsverschieben
AD4E         bsr     FEAD62              ; ---> Kodieren
AD52         move.l  d3,dØ               ; dØ := d3 = Daten-Langwort
AD54         bsr     FEAD62              ; ---> Kodieren
AD58         bsr     FEADBE              ; ---> Grenzkorrektur
AD5C         movem.l (a7)+,d2-d3         ; Register wiederherstellen
AD6Ø         rts
AD62
AD62 FEAD62  andi.l  #55555555,dØ        ; Geradzahlige Bits löschen
AD68         move.l  dØ,d2               ; d2 := Ergebnis
AD6A         eori.l  #55555555,d2        ; Ungeradzahlige Bits flippen
AD7Ø         move.l  d2,d1               ; d1 := Ergebnis
AD72         lsl.l   #1,d2               ; d2 um 1 Bit linksverschieben
AD74         lsr.l   #1,d1               ; d1 um 1 Bit rechtsverschieben
AD76         bset.l  #1F,d1              ; Bit 31 in d1 setzen
AD7A         and.l   d2,d1               ; Alle Ø-Bits von d2 in d1 löschen
AD7C         or.l    d1,dØ               ; Alle 1-Bits von d1 in dØ setzen
AD7E         btst.b  #Ø,-1(aØ)           ; Bit Ø im vorausgehenden Byte = Ø?
AD84         beq.s   FEAD8A              ; ja: ->
AD86         bclr.l  #1F,dØ              ; Bit 31 in dØ setzen
AD8A FEAD8A  move.l  dØ,(aØ)+            ; und dØ speichern
AD8C         rts
AD8E
AD8E ;--------------------------------   64 MFM-Bits ab aØ in dØ dekodieren
AD8E
AD8E FEAD8E  move.l  (aØ)+,dØ            ; dØ := erste 32 Bits
AD9Ø         move.l  (aØ)+,d1            ; d1 := nächste 32 Bits
AD92         andi.l  #55555555,dØ        ; Alle geraden Bits in dØ
AD98         andi.l  #55555555,d1        ; und in d1 löschen
AD9E         lsl.l   #1,dØ               ; dØ um 1 Bit linksverschieben
ADAØ         or.l    d1,dØ               ; d1 einodern
ADA2         rts
ADA4
```

```
ADA4 ;--------------------------------  Prüfsumme berechnen
ADA4
ADA4 FEADA4  move.l  d2,-(a7)          ; d2 retten
ADA6         lsr.w   #2,d1             ; d1 := d1 / 4
ADA8         subq.w  #1,d1             ; - 1
ADAA         moveq   #0,d0             ; d0 := 0
ADAC FEADAC  move.l  (a0)+,d2          ; d2 := nächstes Langwort aus Puffer
ADAE         eor.l   d2,d0             ; mit d0 ex-odern
ADB0         dbra    d1,FEADAC         ; wiederholen bis d1 = -1 ->
ADB4         andi.l  #55555555,d0      ; alle geraden Bits löschen
ADBA         move.l  (a7)+,d2          ; d2 wiederherstellen
ADBC         rts
ADBE
ADBE ;--------------------------------  Korrektur an Bereichsgrenze
ADBE
ADBE FEADBE  move.b  (a0),d0           ; d0 := 1. Byte nach der Grenze
ADC0         btst.b  #0,-1(a0)         ; Letztes Bit vor der Grenze = 1?
ADC6         bne.s   FEADD4            ; ja: ->
ADC8         btst.l  #6,d0             ; 2. Bit nach der Grenze = 1?
ADCC         bne.s   FEADDA            ; ja: Fertig ->
ADCE         bset.l  #7,d0             ; 1. Bit nach der Grenze setzen
ADD2         bra.s   FEADD8            ; --->
ADD4
ADD4 FEADD4  bclr.l  #7,d0             ; 1. Bit nach der Grenze löschen
ADD8 FEADD8  move.b  d0,(a0)           ; Byte zurückspeichern
ADDA FEADDA  rts
ADDC
ADDC ;--------------------------------  Laufwerk anfordern
ADDC
ADDC FEADDC  movem.l a2/a4,-(a7)       ; Register retten
ADE0         movea.l a6,a2             ; a6 in a2 retten
ADE2         movea.l 34(a2),a4         ; a4 := SysBase
ADE6         movea.l 3C(a2),a6         ; a6 -> disk.resource
ADEA FEADEA  lea     D0(a3),a1         ; a1 := UnitPointer (-> Message)
ADEE         jsr     -12(a6)           ; ---> dr_GetUnit
ADF2         tst.l   d0                ; Disk beschäftigt?
ADF4         bne.s   FEAE12            ; nein: ->
ADF6         exg     a4,a6             ; a6 := SysBase, a4 -> disk.resource
ADF8 FEADF8  move.l  #400,d0           ; d0 := Wecksignal
ADFE         jsr     -13E(a6)          ; ---> Wait
```

```
AE02          lea     AE(a3),a0           ; a0 -> ReplyPort
AE06          jsr     -174(a6)            ; ---> GetMsg
AE0A          tst.l   d0                  ; Message vorhanden?
AE0C          beq.s   FEADF8              ; nein: Weiter warten ->
AE0E          exg     a4,a6               ; a6 -> disk.resource, a4 := SysBase
AE10          bra.s   FEADEA              ; ---> neuer Versuch
AE12
AE12 FEAE12   exg     a2,a6               ; a2 -> disk.resource, a6 wiederherst.
AE14          move.b  41(a3),d1           ; d1 := Disk-Kontrollbits
AE18          ori.b   #7F,d1              ; Kontrollbyte := Laufwerk deselektieren
AE1C          move.b  d1,BFD100           ; Kontrollbyte schreiben
AE22          lea     D0(a3),a0           ; a0 -> ReplyMsg
AE26          cmpa.l  d0,a0               ; Neuer Benutzer?
AE28          beq.s   FEAE3C              ; nein: ->
AE2A          lea     DFF000,a0           ; a0 -> ChipBase
AE30          move.w  #600,9E(a0)         ; ADKCON: CLR MSBSYNC, WORDSYNC
AE36          move.w  #9100,9E(a0)        ; ADKCON: SET MFMPREC, FAST
AE3C FEAE3C   movem.l (a7)+,a2/a4         ; Register wiederherstellen
AE40          rts
AE42
AE42 ;--------------------------------   Laufwerk freigeben
AE42
AE42 FEAE42   move.l  a6,-(a7)            ; a6 retten
AE44          movea.l 3C(a6),a6           ; a6 -> disk.resource
AE48          jsr     -18(a6)             ; ---> dr_GiveUnit
AE4C          movea.l (a7)+,a6            ; a6 wiederherstellen
AE4E          rts
AE50
AE50 ;--------------------------------   Trackdisk-Task-Routine
AE50
AE50 FEAE50   movea.l 8(a7),a6            ; a6 := SysBase
AE54          movea.l 4(a7),a3            ; a3 -> MsgPort
AE58          lea     12E(a3),a0          ; a0 -> Task
AE5C          move.l  a0,10(a3)           ; mp_SigTask := a0
AE60          bsr     FE9960              ; ---> Laufwerkprüfung, Zeitsignal
AE64 FEAE64   bsr.s   FEAE7A              ; ---> IORequests bearbeiten
AE66          move.l  #300,d0             ; d0 := Wecksignal
AE6C          move.l  a6,-(a7)            ; a6 retten
AE6E          movea.l 34(a6),a6           ; a6 := SysBase
AE72          jsr     -13E(a6)            ; ---> Wait
```

```
AE76          movea.l (a7)+,a6          ; a6 wiederherstellen
AE78          bra.s   FEAE64            ; ---> Loop
AE7A
AE7A FEAE7A   bset.b  #0,22(a3)         ; Wiederholsperre setzen
AE80          bne     FEAF4A            ; war schon gesetzt: ->
AE84 FEAE84   movea.l a3,a0             ; a0 := a3 -> trackdisk.unit MsgPort
AE86          move.l  a6,-(a7)          ; a6 retten
AE88          movea.l 34(a6),a6         ; a6 := SysBase
AE8C          jsr     -174(a6)          ; ---> GetMsg
AE90          movea.l (a7)+,a6          ; a6 wiederherstellen
AE92          tst.l   d0                ; Message vorhanden?
AE94          beq     FEAF3E            ; nein: Fertig ->
AE98          movea.l d0,a2             ; a2 := d0 -> Message (IORequest)
AE9A          bclr.b  #3,40(a3)         ; Close-Flag löschen
AEA0          beq     FEAF1E            ; war nicht gesetzt: ->
AEA4          movea.l 52(a3),a0         ; a0 -> Spurpuffer
AEA8          bclr.b  #0,2(a0)          ; Änderungsflag löschen
AEAE          beq     FEAEBA            ; war nicht gesetzt: ->
AEB2          move.l  a0,4E(a3)         ; Schreibadresse := a0
AEB6          bsr     FEA958            ; ---> Puffer auf Disk schreiben
AEBA FEAEBA   movea.l 52(a3),a0         ; a0 -> Spurpuffer
AEBE          moveq   #-1,d0            ; d0 := -1
AEC0          move.w  d0,0(a0)          ; Spurnummer im Puffer ungültig machen
AEC4          move.w  d0,4C(a3)         ; Spurnummer in Unit ungültig machen
AEC8          moveq   #0,d0             ; d0 := 0
AECA          bsr     FEA462            ; ---> Motor ausschalten
AECE          movea.l a6,a0             ; a0 := a6 -> trackdisk.device
AED0          movea.l 34(a0),a6         ; a6 := SysBase
AED4          addq.b  #1,127(a6)        ; TDNestCnt erhöhen: Kein Taskwechsel
AED8          movea.l a0,a6             ; a6 := a0 -> trackdisk.device
AEDA          tst.w   24(a3)            ; Open-Zähler testen
AEDE          bne     FEAF12            ; Device aktiv: ->
AEE2          moveq   #0,d0             ; d0 := 0
AEE4          move.b  43(a3),d0         ; d0 := UnitNum
AEE8          move.l  a6,-(a7)          ; a6 retten
AEEA          movea.l 3C(a6),a6         ; a6 -> disk.resource
AEEE          jsr     -C(a6)            ; ---> dr_FreeUnit
AEF2          movea.l (a7)+,a6          ; a6 wiederherstellen
AEF4          lea     24(a6),a0         ; a0 -> UnitPointer-Bereich
AEF8          moveq   #0,d0             ; d0 := 0
```

```
AEFA          move.b  43(a3),d0          ; d0 := UnitNum
AEFE          lsl.l   #2,d0              ; mal 4
AF00          adda.l  d0,a0              ; a0 -> UnitPointer
AF02          clr.l   (a0)               ; UnitPointer löschen
AF04          suba.l  a1,a1              ; a1 := 0: Diese Task
AF06          move.l  a6,-(a7)           ; a6 retten
AF08          movea.l 34(a6),a6          ; a6 := SysBase
AF0C          jsr     -120(a6)           ; ---> RemTask
AF10          movea.l (a7)+,a6           ; a6 wiederherstellen
AF12 FEAF12   move.l  a6,-(a7)           ; a6 retten
AF14          movea.l 34(a6),a6          ; a6 := SysBase
AF18          jsr     -8A(a6)            ; ---> Permit
AF1C          movea.l (a7)+,a6           ; a6 wiederherstellen
AF1E FEAF1E   movea.l a2,a1              ; a1 := a2 -> Message
AF20          lea     86(a3),a0          ; a0 -> IORequest unit_VBlank
AF24          cmpa.l  a0,a2              ; Beide gleich?
AF26          bne.s   FEAF30             ; nein: ->
AF28          bsr     FE9960             ; ---> Laufwerkprüfung, Zeitsignal
AF2C          bra     FEAE84             ; ---> Loop
AF30
AF30 FEAF30   bset.b  #1,22(a3)          ; Kommandoflag setzen
AF36          bsr     FEA01C             ; ---> Kommando-Verteiler
AF3A          bra     FEAE84             ; ---> Loop
AF3E
AF3E FEAF3E   bclr.b  #1,22(a3)          ; Kommandoflag löschen
AF44          bclr.b  #0,22(a3)          ; Wiederholsperre löschen
AF4A FEAF4A   rts
AF4C
AF4C ;--------------------------------    Sektor-Header suchen
AF4C
AF4C FEAF4C   movem.l d2-d4/a2,-(a7)     ; Register retten
AF50          move.w  #AAAA,d3           ; d3 := %1010101010101010
AF54          move.w  #5555,d4           ; d4 := %0101010101010101
AF58          movea.l a0,a2              ; a2 := a0 -> Suchanfang
AF5A          adda.l  d0,a2              ; a2 -> Suchende
AF5C FEAF5C   move.w  (a0)+,d2           ; d2 := nächstes Wort
AF5E          cmp.w   d3,d2              ; = $AAAA?
AF60          beq.s   FEAF98             ; ja: ->
AF62          cmp.w   d4,d2              ; = $5555?
AF64          beq.s   FEAF70             ; ja: ->
```

```
AF66          cmpa.l  a0,a2               ; Suchende erreicht?
AF68          bhi.s   FEAF5C              ; nein: Loop ->
AF6A FEAF6A   moveq   #-1,d0              ; d0 := -1
AF6C          movea.l d0,a0               ; a0 := d0 = -1
AF6E          bra.s   FEAF92              ; ---> Abschluß
AF70
AF70 FEAF70   moveq   #F,d0               ; d0 := 15
AF72          lea     FEAFA2,a1           ; a1 -> Tabelle 1
AF78 FEAF78   cmpa.l  a0,a2               ; Suchende erreicht?
AF7A          bls.s   FEAF6A              ; ja: a0 := -1, Abschluß ->
AF7C          move.w  (a0)+,d1            ; d1 := nächstes Wort
AF7E          cmp.w   d2,d1               ; = $AAAA bzw. =$5555?
AF80          beq.s   FEAF78              ; ja: weiterlesen ->
AF82          subq.l  #2,a0               ; a0 auf letztes Wort zurücksetzen
AF84          move.l  (a0),d1             ; d1 := erstes verschiedenes Langwort
AF86 FEAF86   cmp.l   (a1)+,d1            ; Langwort mit Tabelle vergleichen
AF88          beq.s   FEAF90              ; gefunden: ->
AF8A          subq.l  #2,d0               ; Zähler d0 um 2 vermindern
AF8C          bge.s   FEAF86              ; noch nicht negativ: weitersuchen ->
AF8E          bra.s   FEAF5C              ; --->
AF90
AF90 FEAF90   subq.l  #4,a0               ; a0 um 1 Langwort zurücksetzen
AF92 FEAF92   movem.l (a7)+,d2-d4/a2      ; Register wiederherstellen
AF96          rts
AF98
AF98 FEAF98   moveq   #E,d0               ; d0 := 14
AF9A          lea     FEAFC2,a1           ; a1 -> Tabelle 2
AFA0          bra.s   FEAF78              ; --->
AFA2
AFA2 ;------ Tabelle der möglichen Anfangs-Bitmuster
AFA2
AFA2 FEAFA2   DC.L    2244A244
AFA6          DC.L    48912891
AFAA          DC.L    52244A24
AFAE          DC.L    54891289
AFB2          DC.L    552244A2
AFB6          DC.L    55489128
AFBA          DC.L    5552244A
AFBE          DC.L    55548912
AFC2
```

```
AFC2 FEAFC2  DC.L    91225122
AFC6         DC.L    A4489448
AFCA         DC.L    A9122512
AFCE         DC.L    AA448944
AFD2         DC.L    AA912251
AFD6         DC.L    AAA44894
AFDA         DC.L    AAA91225
AFDE         DC.L    44894489
AFE2
AFE2 ;--------------------------------   Spur normalisieren und prüfen
AFE2
AFE2 FEAFE2  movem.l d2-d6/a2,-(a7)      ; Register retten
AFE6         link    a4,#-10             ; Stackrahmen von 16 Bytes einrichten
AFEA         movea.l 4E(a3),a2           ; a2 -> Spurpuffer
AFEE         lea     680(a2),a2          ; a2 -> Spurpuffer + 1664
AFF2         movea.l a2,a0               ; a0 := a2
AFF4         addq.l  #2,a0               ; + 2
AFF6         move.l  #ABC,d0             ; d0 := 2748
AFFC         bsr     FEAF4C              ; ---> SektorHeader suchen
B000         cmpa.l  #-1,a0              ; gefunden?
B006         beq     FEB1E8              ; nein: tderr_NoSecHdr ->
B00A         move.l  a0,d5               ; d5 := a0 -> d2 Bits vor Sektoranfang
B00C         move.l  d0,d2               ; d2 := d0 = Bit-Verschiebezahl
B00E         addq.l  #8,a0               ; a0 -> d2 Bits vor Header
B010         moveq   #9,d4               ; d4 := 9 = Zähler für Prüfsumme
B012         moveq   #0,d6               ; d6 := 0 = Register für Prüfsumme
B014         tst.l   d2                  ; Bit-Verschiebung
B016         bne.s   FEB03C              ; nicht 0: ->
B018         move.l  (a0),-8(a4)         ; Header in Stackrahmen kopieren
B01C         move.l  4(a0),-4(a4)
B022         move.l  #55555555,d1        ; d1 := Löschmaske für gerade Bits
B028 FEB028  move.l  (a0)+,d0            ; Nächstes Langwort lesen
B02A         and.l   d1,d0               ; Gerade Bits löschen
B02C         eor.l   d0,d6               ; Prüfsumme berechnen
B02E         dbra    d4,FEB028           ; wiederholen bis d4 = -1 ->
B032         move.l  (a0)+,-10(a4)       ; Header-Prüfsumme in Stackrahmen
B036         move.l  (a0),-C(a4)
B03A         bra.s   FEB070              ; --->
B03C
B03C FEB03C  bsr     FEB204              ; ---> Bitverschiebung
```

```
B04Ø          move.l  dØ,-8(a4)          ; 1. Headerlangwort in Stackrahmen
B044          bsr     FEB2Ø4             ; ---> Bitverschiebung
B048          move.l  dØ,-4(a4)          ; 2. Headerlangwort in Stackrahmen
B04C          movea.l d5,aØ              ; aØ wiederherstellen
B04E          addq.l  #8,aØ              ; auf nächstes Langwort setzen
B05Ø FEBØ5Ø   bsr     FEB2Ø4             ; ---> Bitverschiebung
B054          andi.l  #55555555,dØ       ; Gerade Bits löschen
B05A          eor.l   dØ,d6              ; Prüfsumme berechnen
B05C          dbra    d4,FEBØ5Ø          ; wiederholen bis d4 = -1 ->
B06Ø          bsr     FEB2Ø4             ; ---> Bitverschiebung
B064          move.l  dØ,-1Ø(a4)         ; 1. LW Header-Prüfsumme in Stackrahmen
B068          bsr     FEB2Ø4             ; ---> Bitverschiebung
B06C          move.l  dØ,-C(a4)          ; 2. LW Header-Prüfsumme in Stackrahmen
B07Ø FEBØ7Ø   lea     -1Ø(a4),aØ         ; aØ -> Prüfsumme
B074          bsr     FEAD8E             ; ---> dØ := 64 MFM-Bits ab aØ dekodiert
B078          cmp.l   dØ,d6              ; Prüfsumme richtig?
B07A          bne     FEB1EC             ; nein: tderr_BadSecHdr ->
B07E          lea     -8(a4),aØ          ; aØ -> Header
B082          bsr     FEAD8E             ; ---> dØ := 64 MFM-Bits ab aØ dekodiert
B086          move.l  dØ,d3              ; d3 := dØ = Header
B088          move.l  d3,-(a7)           ; auf Stack legen
B08A          cmpi.b  #FF,Ø(a7)          ; 1. Byte Formatbyte?
B09Ø          bne     FEB1EC             ; nein: tderr_BadSecHdr ->
B094          move.b  1(a7),dØ           ; dØ := 2. Byte
B098          cmp.b   4B(a3),dØ          ; 2. Byte = Spurnummer?
B09C          bne     FEB1EC             ; nein: tderr_BadSecHdr ->
B0AØ          move.l  (a7)+,d3           ; d3 := Header vom Stack
B0A2          moveq   #Ø,dØ              ; dØ := Ø
B0A4          move.b  d3,dØ              ; dØ := Sektoroffset zur Lücke
B0A6          mulu    #44Ø,dØ            ; * 1Ø88 = Byteoffset zur Lücke
B0AA          movea.l d5,aØ              ; aØ wiederherstellen
B0AC          movea.l a2,a1              ; a1 := a2 -> Spurpuffer + 1664
B0AE          move.l  d2,d1              ; d1 := d2 = Bitverschiebung
B0BØ          move.l  dØ,d4              ; d4 := dØ = Byteoffset zur Lücke
B0B2          bsr     FEB214             ; ---> Spur bis zur Lücke normalisieren
B0B6          moveq   #Ø,d2              ; d2 := Ø
B0B8          move.b  d3,d2              ; d2 := d2 := Byteoffset zur Lücke
B0BA          subi.l  #B,d2              ; - 11 (Zahl der Sektoren pro Spur)
B0CØ          neg.l   d2                 ; d2 := Zahl der Sektoren hinter Lücke
B0C2          beq.s   FEBØEE             ; Lücke am Ende der Spur: ->
```

```
B0C4          add.l   d4,d5                ; d5 -> Anfang der Lücke
B0C6          move.l  #67C,d0              ; d0 := 1660
B0CC          movea.l d5,a0                ; a0 -> Anfang der Lücke
B0CE          addq.l  #2,a0                ; + 2
B0D0          bsr     FEAF4C               ; ---> SektorHeader suchen
B0D4          cmpa.l  #-1,a0               ; gefunden?
B0DA          beq     FEB200               ; nein: tderr_TooFewSecs ->
B0DE          move.l  d0,d1                ; d1 := d0 = Bitverschiebung
B0E0          movea.l a2,a1                ; a1 := a2 -> Spurpuffer + 1664
B0E2          adda.l  d4,a1                ; a1 -> Anfang der Lücke
B0E4          move.l  d2,d0                ; d0 := d2 = Sektorzahl hinter Lücke
B0E6          mulu    #440,d0              ; * 1088 = Bytezahl hinter Lücke
B0EA          bsr     FEB214               ; ---> Restspur normalisieren
B0EE FEB0EE   movea.l a2,a0                ; a0 := a2 -> Spurpuffer + 1664
B0F0          adda.l  d4,a0                ; a0 -> Sektor 1
B0F2          bsr     FEADBE               ; ---> Grenzkorrektur
B0F6          lea     2EC0(a2),a0          ; a0 := erste Adresse hinter der Spur
B0FA          move.w  #AAA8,d0             ; d0 := $AAA8
B0FE          btst.b  #0,-1(a0)            ; Letztes Bit vor Spurende gesetzt?
B104          beq     FEB10C               ; nein: ->
B108          bclr.l  #F,d0                ; d0 := $2AA8
B10C FEB10C   move.w  d0,(a0)              ; d0 hinter Spurende setzen
B10E          move.w  #AAAA,(a2)           ; $AAAA an Spuranfang setzen
B112          moveq   #0,d4                ; d4 := 0 = Abstand vom Spuranfang
B114          moveq   #B,d2                ; d2 := 11 = Sektoroffset zum Spurende
B116          move.w  d3,d5                ; d5 := d3 = 2. Wort im 1. Header
B118          lsr.w   #8,d5                ; d5 := Sektornummer
B11A FEB11A   cmpi.l  #2AAAAAAA,0(a2,d4.w)  ; Richtiger Sektoranfang?
B122          beq.s   FEB130               ; ja: ->
B124          cmpi.l  #AAAAAAAA,0(a2,d4.w)  ; Richtiger Sektoranfang?
B12C          bne     FEB1F0               ; nein: tderr_BadSecPreamble ->
B130 FEB130   cmpi.l  #44894489,4(a2,d4.w)  ; Folgen die richtigen Sync-Worte?
B138          bne     FEB1F0               ; nein: tderr_BadSecPreamble ->
B13C          lea     8(a2,d4.w),a0        ; a0 -> Header
B140          moveq   #28,d1               ; d1 := 40 Bytes
B142          bsr     FEADA4               ; ---> Prüfsumme berechnen
B146          move.l  d0,d6                ; d6 := d0 = Prüfsumme
B148          lea     30(a2,d4.w),a0       ; a0 -> Header-Prüfsumme
B14C          bsr     FEAD8E               ; ---> d0 := 64 MFM-Bits ab a0 dekodiert
B150          cmp.l   d6,d0                ; Prüfsumme richtig?
```

```
B152          bne     FEB1F8              ; nein: tderr_BadHdrSum ->
B156          lea     8(a2,d4.w),aØ       ; aØ -> Header
B15A          bsr     FEAD8E              ; ---> dØ := 64 MFM-Bits ab aØ dekodiert
B15E          move.l  dØ,-(a7)            ; Header auf Stack legen
B16Ø          cmpi.b  #FF,Ø(a7)           ; Formatbyte richtig?
B166          bne     FEB1F4              ; nein: tderr_BadSecID ->
B16A          move.b  1(a7),d1            ; d1 := Spurnummer
B16E          cmp.b   4B(a3),d1           ; Spurnummer richtig?
B172          bne     FEB1F4              ; nein: tderr_BadSecID ->
B176          move.b  2(a7),d1            ; d1 := Sektornummer
B17A          cmp.b   d5,d1               ; Sektornummer richtig?
B17C          bne     FEB1F4              ; nein: tderr_BadSecID ->
B18Ø          move.b  d2,3(a7)            ; Sektoroffset in Header eintragen
B184          move.l  (a7)+,dØ            ; dØ := Header
B186          lea     8(a2,d4.w),aØ       ; aØ -> Header
B18A          bsr     FEAD46              ; ---> dØ kodieren und ab aØ speichern
B18E          lea     8(a2,d4.w),aØ       ; aØ -> Header
B192          moveq   #28,d1              ; d1 := 4Ø Bytes
B194          bsr     FEADA4              ; ---> Prüfsumme für Header berechnen
B198          lea     3Ø(a2,d4.w),aØ      ; aØ -> Header-Prüfsumme
B19C          bsr     FEAD46              ; ---> dØ kodieren und ab aØ speichern
B1AØ          lea     4Ø(a2,d4.w),aØ      ; aØ -> Datenblock
B1A4          move.w  #4ØØ,d1             ; d1 := 1Ø24 Bytes
B1A8          bsr     FEADA4              ; ---> Prüfsumme für Daten berechnen
B1AC          move.l  dØ,d6               ; d6 := dØ = Prüfsumme
B1AE          lea     38(a2,d4.w),aØ      ; aØ -> Block-Prüfsumme
B1B2          bsr     FEAD8E              ; ---> dØ := 64 MFM-Bits ab aØ dekodiert
B1B6          cmp.l   d6,dØ               ; Prüfsumme richtig?
B1B8          bne     FEB1FC              ; nein: tderr_BadSecSum ->
B1BC          subq.l  #1,d2               ; Sektoroffset vermindern
B1BE          addq.b  #1,d5               ; Sektornummer erhöhen
B1CØ          cmpi.b  #B,d5               ; Sektornummer < 11?
B1C4          blt.s   FEB1C8              ; ja: ->
B1C6          moveq   #Ø,d5               ; Sektornummer := Ø
B1C8 FEB1C8   addi.w  #44Ø,d4             ; d4 auf nächsten Sektoranfang setzen
B1CC          cmpi.w  #2ECØ,d4            ; Spurende erreicht?
B1DØ          bne     FEB11A              ; nein: Loop ->
B1D4          move.l  d3,d1               ; d1 := d3 = 1. SektorHeader
B1D6          lsr.l   #8,d1               ; d1.b := Nummer des 1. Sektors
B1D8          moveq   #Ø,dØ               ; dØ := Ø
```

```
B1DA          move.b  d1,dØ               ; dØ := Nummer des 1. Sektors
B1DC FEB1DC   unlk    a4                  ; Stackrahmen freigeben
B1DE          movem.l (a7)+,d2-d6/a2      ; Register wiederherstellen
B1E2          rts
B1E4
B1E4 ;------ Fehlerausgänge
B1E4
B1E4 FEB1E4   nop
B1E6          bra.s   FEB1DC              ; ---> Abschluß
B1E8
B1E8 FEB1E8   moveq   #15,dØ              ; dØ := tderr_NoSecHdr
B1EA          bra.s   FEB1E4              ; ---> Abschluß
B1EC
B1EC FEB1EC   moveq   #1B,dØ              ; dØ := tderr_BadSecHdr
B1EE          bra.s   FEB1E4              ; ---> Abschluß
B1FØ
B1FØ FEB1FØ   moveq   #16,dØ              ; dØ := tderr_BadSecPreamble
B1F2          bra.s   FEB1E4              ; ---> Abschluß
B1F4
B1F4 FEB1F4   moveq   #17,dØ              ; dØ := tderr_BadSecID
B1F6          bra.s   FEB1E4              ; ---> Abschluß
B1F8
B1F8 FEB1F8   moveq   #18,dØ              ; dØ := tderr_BadHdrSum
B1FA          bra.s   FEB1E4              ; ---> Abschluß
B1FC
B1FC FEB1FC   moveq   #19,dØ              ; dØ := tderr_BadSecSum
B1FE          bra.s   FEB1E4              ; ---> Abschluß
B2ØØ
B2ØØ FEB2ØØ   moveq   #1A,dØ              ; dØ := tderr_TooFewSecs
B2Ø2          bra.s   FEB1E4              ; ---> Abschluß
B2Ø4
B2Ø4 ;--------------------------------   Bitverschiebung
B2Ø4
B2Ø4 FEB2Ø4   move.l  (aØ)+,dØ            ; dØ := Langwort aus Spur
B2Ø6          move.w  (aØ),d1             ; d1 := nächstes Wort
B2Ø8          moveq   #1Ø,d3              ; d3 := 16
B2ØA          sub.l   d2,d3               ; - d2
B2ØC          lsl.l   d3,dØ               ; dØ um d3 Bits linksverschieben
B2ØE          lsr.w   d2,d1               ; d1 um d2 Bits rechtsverschieben
```

```
B21Ø          or.w    d1,dØ               ; und einodern
B212          rts
B214
B214 ;------ dØ Bytes ab Adresse aØ um d1 Bits verschieben und nach a1 kopieren
B214
B214 FEB214   link    a2,#-1E             ; Stackrahmen 3Ø Bytes für BltNode
B218          move.b  d1,-8(a2)           ; Bitverschiebung eintragen
B21C          tst.l   d1                  ; Bitverschiebung = Ø?
B21E          beq.s   FEB222              ; ja: ->
B22Ø          addq.l  #2,dØ               ; Byteoffset um 2 erhöhen
B222 FEB222   move.l  dØ,d1               ; d1 := dØ = Byteoffset zur Lücke
B224          addi.l  #3F,d1              ; + 63 zur Aufrundung
B22A          andi.w  #FFCØ,d1            ; BltSize Höhe := Byteoffset \ 64
B22E          ori.w   #2Ø,d1              ; BltSize Breite := 32 Worte = 64 Bytes
B232          move.w  d1,-A(a2)           ; BLTSIZE-Wert eintragen
B236          movem.l dØ/aØ-a1,-16(a2)    ; Bytezahl, Quelle, Ziel eintragen
B23C          move.l  #FEB26Ø,-1A(a2)     ; Blitterfunktion eintragen
B244          move.l  a3,-4(a2)           ; trackdisk.unit eintragen
B248          lea     -1E(a2),a1          ; a1 -> BltNode
B24C          move.l  a6,-(a7)            ; a6 retten
B24E          movea.l 38(a6),a6           ; a6 := GfxBase
B252          jsr     -114(a6)            ; ---> QBlit
B256          movea.l (a7)+,a6            ; a6 wiederherstellen
B258          bsr     FEA7ØA              ; ---> Auf Rückmeldung warten
B25C          unlk    a2                  ; Stackrahmen freigeben
B25E          rts
B26Ø
B26Ø ;------ Blitter-Funktionen
B26Ø
B26Ø FEB26Ø   move.l  a5,-(a7)            ; a5 retten
B262          movea.l a1,a5               ; a5 := a1 -> BltNode
B264          bsr     FEB2CC              ; ---> Masken und Modulo setzen
B268          move.b  16(a5),dØ           ; dØ := Bitverschiebung
B26C          moveq   #C,d1               ; d1 := 12
B26E          lsl.w   d1,dØ               ; dØ um 12 Bits linksverschieben
B27Ø          move.w  #5CC,4Ø(aØ)         ; BLTCONØ := USEB,D+LF7,6,3,2
B276          move.w  dØ,42(aØ)           ; BLTCON1 := Bitverschiebung
B27A          movem.l 8(a5),dØ-d1/a1      ; dØ := Bytes, d1 -> Quelle, a1 -> Ziel
B28Ø          move.l  d1,4C(aØ)           ; BLTBPNT := d1 (Quelle B)
B284          move.l  #FEB2B8,4(a5)       ; Nächste Funktion in BltNode eintragen
```

```
B28C          tst.b   16(a5)              ; Bitverschiebung = Ø?
B29Ø          beq.s   FEB29C              ; ja: ->
B292          subq.l  #2,a1               ; Ziel um 2 vermindern
B294          move.l  #FEB2AE,4(a5)       ; Nächste Funktion in BltNode eintragen
B29C FEB29C   move.l  a1,54(aØ)           ; BLTDPNT := a1 (Ziel D)
B2AØ          move.w  (a1),18(a5)         ; Wort an Zielanfang merken
B2A4          move.w  14(a5),58(aØ)       ; BLTSIZE schreiben startet Blitter
B2AA          movea.l (a7)+,a5            ; a5 wiederherstellen
B2AC          rts
B2AE
B2AE FEB2AE   movea.l 1Ø(a1),aØ           ; aØ -> Ziel
B2B2          move.w  18(a1),-2(aØ)       ; gemerktes Wort davorsetzen
B2B8 FEB2B8   moveq   #Ø,dØ               ; dØ := Ø
B2BA          move.l  dØ,4(a1)            ; Keine weitere Funktion
B2BE          movea.l 1A(a1),a1           ; a1 -> trackdisk.unit
B2C2          bsr     FEA6F2              ; ---> Rückmeldung senden
B2C6          moveq   #Ø,dØ               ; dØ := Ø
B2C8          rts
B2CA
B2CA          DC.W    Ø
B2CC
B2CC ;--------------------------------   Blitter-Masken und Modulo setzen
B2CC
B2CC FEB2CC   moveq   #Ø,dØ               ; dØ := Ø
B2CE          lea     44(aØ),a1           ; a1 -> BLTAFWM
B2D2          move.l  #FFFFFFFF,(a1)      ; Maskierung beseitigen
B2D8          lea     62(aØ),a1           ; a1 -> BLTBMOD
B2DC          move.l  dØ,(a1)+            ; Alle Modulos löschen
B2DE          move.w  dØ,(a1)+
B2EØ          addq.l  #8,a1               ; a1 -> BLTCDAT
B2E2          move.w  #5555,(a1)
B2E6          rts
B2E8
B2E8 ;--------------------------------   td_RawRead
B2E8
B2E8 FEB2E8   lea     -DC2(pc),aØ         ; aØ := FEA528 -> Leseroutine
B2EC          bra.s   FEB2F2              ; --->
B2EE
```

```
B2EE ;--------------------------------   td_RawWrite
B2EE
B2EE FEB2EE  lea     -D38(pc),aØ          ; aØ := FEA5B8 -> Schreibroutine
B2F2 FEB2F2  movem.l d2-d3/a2-a4,-(a7)    ; Register retten
B2F6         link    a5,#-8               ; Stackrahmen von 8 Bytes einrichten
B2FA         movea.l a1,a2                ; a2 := a1 -> IORequest
B2FC         movea.l aØ,a4                ; a4 := aØ -> Lese- oder Schreibroutine
B2FE         moveq   #1,dØ                ; dØ := 1
B3ØØ         bsr     FEA462               ; ---> Motor einschalten
B3Ø4         move.l  2C(a2),dØ            ; dØ := io_Offset = Spurnummer
B3Ø8         cmp.l   36(a3),dØ            ; dØ >= Zahl der Spuren pro Disk?
B3ØC         bcc.s   FEB35C               ; ja: Fehler ->
B3ØE         bsr     FEA3DA               ; ---> Kopf auf Spur dØ setzen
B312         move.b  dØ,1F(a2)            ; io_Error := RÜckmeldung
B316         bne.s   FEB34E               ; Fehler: ->
B318         move.l  24(a2),dØ            ; dØ := io_Length
B31C         cmp.l   8ØØØ,dØ              ; dØ > 32767?
B322         bcc.s   FEB35C               ; ja: Fehler ->
B324         movea.l 28(a2),aØ            ; aØ := io_Data
B328         lea     -D8Ø(pc),a1          ; a1 := FEA5AA -> DMA-Startroutine
B32C         btst.b  #4,1E(a2)            ; io_Flags iotd_IndexSync gesetzt?
B332         beq.s   FEB348               ; nein: ->
B334         lea     -8(a5),a1            ; a1 -> Stackrahmen
B338         move.l  a1,d1                ; d1 := a1
B33A         lea     52(pc),a1            ; a1 := FEB38E -> Interrupt-Routine
B33E         movem.l d1/a1,11E(a3)        ; is_Data und is_Code eintragen
B344         lea     1E(pc),a1            ; a1 := FEB364 -> Set&Able FLG-Interrupt
B348 FEB348  jsr     (a4)                 ; ---> Lese-/Schreibroutine
B34A         move.b  dØ,1F(a2)            ; io_Error := RÜckmeldung
B34E FEB34E  movea.l a2,a1                ; a1 := a2 -> IORequest
B35Ø         bsr     FEA1BØ               ; ---> IORequest beantworten
B354         unlk    a5                   ; Stackrahmen freigeben
B356         movem.l (a7)+,d2-d3/a2-a4    ; Register wiederherstellen
B35A         rts
B35C
B35C FEB35C  move.b  #FC,1F(a2)           ; io_Error := ioerr_BadLength
B362         bra.s   FEB34E               ; ---> Abschluß
B364
B364 FEB364  movea.l 11E(a3),aØ           ; aØ := is_Data -> Stackrahmen
B368         movem.l dØ/a6,(aØ)           ; io_Length und io_Device eintragen
```

```
B36C          moveq   #10,d0              ; d0 := ICR-Maske 'Reset FLG-Bit'
B36E          move.l  a6,-(a7)            ; a6 retten
B370          movea.l 5E(a6),a6           ; a6 -> ciab.resource
B374          jsr     -18(a6)             ; ---> SetICR
B378          movea.l (a7)+,a6            ; a6 wiederherstellen
B37A          move.l  #90,d0              ; d0 := ICR-Maske 'Set FLG-Bit'
B380          move.l  a6,-(a7)            ; a6 retten
B382          movea.l 5E(a6),a6           ; a6 -> ciab.resource
B386          jsr     -12(a6)             ; ---> AbleICR
B38A          movea.l (a7)+,a6            ; a6 wiederherstellen
B38C          rts
B38E
B38E ;------ IndexSync-Interrupt
B38E
B38E FEB38E   move.l  a2,-(a7)            ; a2 retten
B390          movea.l a1,a2               ; a2 := a1 -> Stackrahmen
B392          move.l  (a2),d0             ; d0 := io_Length
B394          lea     DFF000,a1           ; a1 := ChipBase
B39A          bsr     FEA5AA              ; ---> DMA-Startroutine
B39E          movea.l 4(a2),a0            ; a0 -> trackdisk.device
B3A2          movea.l 5E(a0),a0           ; a0 -> ciab.resource
B3A6          moveq   #10,d0              ; d0 := ICR-Maske 'Reset FLG-Bit'
B3A8          move.l  a6,-(a7)            ; a6 retten
B3AA          movea.l a0,a6               ; a6 -> ciab.resource
B3AC          jsr     -12(a6)             ; ---> AbleICR
B3B0          movea.l (a7)+,a6            ; a6 wiederherstellen
B3B2          movea.l (a7)+,a2            ; a2 wiederherstellen
B3B4          rts
B3B6
B3B6 ;--------------------------------   td_GetDriveType
B3B6
B3B6 FEB3B6   moveq   #0,d0               ; d0 := 0
B3B8          move.b  35(a3),d0           ; d0 := Typ
B3BC          move.l  d0,20(a2)           ; io_Actual := Typ
B3C0          movea.l a2,a1               ; a1 := a2 -> IORequest
B3C2          bsr     FEA1B0              ; ---> IORequest beantworten
B3C6          rts
B3C8
```

```
B3C8 ;--------------------------------   td_GetNumTracks
B3C8
B3C8 FEB3C8  moveq   #0,d0               ; d0 := 0
B3CA         move.w  36(a3),d0           ; d0 := Zahl der Spuren pro Disk
B3CE         move.l  d0,20(a2)           ; io_Actual := d0
B3D2         movea.l a2,a1               ; a1 := a2 -> IORequest
B3D4         bsr     FEA1B0              ; ---> IORequest beantworten
B3D8         rts
B3DA
B3DA         DC.W    0
B3DC
B3DC ***************************************************************************
```

Ciaa.resource Datenbereich

```
-018 *************** ciaa.resource ***************************************
-018
-018          jmp     FC4728              ; ---> SetICR
-012          jmp     FC470A              ; ---> AbleICR
-00C          jmp     FC46E6              ; ---> RemICRVector
-006          jmp     FC4690              ; ---> AddICRVector
0000
0000 ;--------------------------------   Library-Block
0000
0000          DC.L    1B18                ; ln_Succ
0004          DC.L    1A18                ; ln_Pred
0008          DC.B    8                   ; ln_Type (8 = Resource)
0009          DC.B    0                   ; ln_Pri
000A          DC.L    FC4534              ; ln_Name -> 'ciaa.resource'
000E
000E          DC.B    0                   ; lib_Flags
000F          DC.B    0                   ; lib_Pad
0010          DC.W    18                  ; lib_NegSize
0012          DC.W    7C                  ; lib_PosSize
0014          DC.W    0                   ; lib_Version
0016          DC.W    0                   ; lib_Revision
0018          DC.L    0                   ; lib_IdString
001C          DC.L    0                   ; lib_Sum
0020          DC.W    0                   ; lib_OpenCnt
0022
0022 ;--------------------------------   Verschiedene Parameter
0022
0022          DC.L    BFE001              ; ciaa Basisadresse
0026
0026          DC.W    8                   ; cr_IntMask für Interrupt 3
```

```
0028          DC.B    A                   ; cr_IEnable
0029          DC.B    14                  ; cr_IActive
002A
002A ;--------------------------------   Interrupt-Struktur
002A
002A     00   DC.L    08EC                ; ln_Succ
002E     04   DC.L    08E8                ; ln_Pred
0032     08   DC.B    2                   ; ln_Type (2 = Interrupt)
0033     09   DC.B    0                   ; ln_Pri
0034     0A   DC.L    FC4534              ; ln_Name -> 'ciaa.resource'
0038     0E   DC.L    *0000               ; is_Data -> ciaa.resource
003C     12   DC.L    FC461C              ; is_Code -> InterruptServer CIAA
0040
0040 ;--------------------------------   Timer A InterruptVektor
0040
0040          DC.L    0                   ; iv_Data
0044          DC.L    0                   ; iv_Code
0048          DC.L    0                   ; iv_Node
004C
004C ;--------------------------------   Timer B InterruptVektor
004C
004C          DC.L    2F08                ; iv_Data -> timer.device Unit_MicroHz
0050          DC.L    FE9726              ; iv_Code -> IntHandler Unit_MicroHz
0054          DC.L    2F32                ; iv_Node -> Interrupt-Struktur
0058
0058 ;--------------------------------   Alarm InterruptVektor
0058
0058          DC.L    0                   ; iv_Data
005C          DC.L    0                   ; iv_Code
0060          DC.L    0                   ; iv_Node
0064
0064 ;--------------------------------   SerialPort InterruptVektor
0064
0064          DC.L    2C1C                ; iv_Data -> keyboard.device
0068          DC.L    FE571C              ; iv_Code -> kbd_InputHandler
006C          DC.L    2C76                ; iv_Node -> Interrupt-Struktur
0070
```

```
0070 ;--------------------------------   FLAG InterruptVektor
0070
0070          DC.L    05A314              ; iv_Data -> parallel.device
0074          DC.L    06232E              ; iv_Code -> pd_IntHandler
0078          DC.L    05A34A              ; iv_Node -> Interrupt-Struktur
007C
007C ************************************************************************
```

Ciab.resource Datenbereich

```
-018 *************** ciab.resource ****************************************
-018
-018          jmp     FC4728              ; ---> SetICR
-012          jmp     FC470A              ; ---> AbleICR
-00C          jmp     FC46E6              ; ---> RemICRVector
-006          jmp     FC4690              ; ---> AddICRVector
0000
0000 ;--------------------------------   Library-Block
0000
0000          DC.L    1EDE                ; ln_Succ
0004          DC.L    1A80                ; ln_Pred
0008          DC.B    8                   ; ln_Type (8 = Resource)
0009          DC.B    0                   ; ln_Pri
000A          DC.L    FC4542              ; ln_Name -> 'ciab.resource'
000E
000E          DC.B    0                   ; lib_Flags
000F          DC.B    0                   ; lib_Pad
0010          DC.W    18                  ; lib_NegSize
0012          DC.W    7C                  ; lib_PosSize
0014          DC.W    0                   ; lib_Version
0016          DC.W    0                   ; lib_Revision
0018          DC.L    0                   ; lib_IdString
001C          DC.L    0                   ; lib_Sum
0020          DC.W    0                   ; lib_OpenCnt
0022
0022 ;--------------------------------   Verschiedene Parameter
0022
0022          DC.L    BFD000              ; ciab Basisadresse
0026
0026          DC.W    2000                ; cr_IntMask für Interrupt 13
```

```
0028          DC.B    4                 ; cr_IEnable
0029          DC.B    10                ; cr_IActive
002A
002A ;--------------------------------  Interrupt-Struktur
002A
002A     00   DC.L    092E              ; ln_Succ
002E     04   DC.L    092A              ; ln_Pred
0032     08   DC.B    2                 ; ln_Type (2 = Interrupt)
0033     09   DC.B    0                 ; ln_Pri
0034     0A   DC.L    FC4542            ; ln_Name -> 'ciab.resource'
0038     0E   DC.L    *0000             ; is_Data
003C     12   DC.L    FC4610            ; is_Code
0040
0040 ;--------------------------------  Timer A InterruptVector
0040
0040          DC.L    0                 ; iv_Data
0044          DC.L    0                 ; iv_Code
0048          DC.L    0                 ; iv_Node
004C
004C ;--------------------------------  Timer B InterruptVektor
004C
004C          DC.L    0                 ; iv_Data
0050          DC.L    0                 ; iv_Code
0054          DC.L    0                 ; iv_Node
0058
0058 ;--------------------------------  Alarm InterruptVektor
0058
0058          DC.L    21FE              ; iv_Data -> graphics.library
005C          DC.L    FC6D68            ; iv_Code -> AlarmInterruptHandler
0060          DC.L    225E              ; iv_Node -> gb_TimSrv
0064
0064 ;--------------------------------  SerialPort InterruptVektor
0064
0064          DC.L    0                 ; iv_Data
0068          DC.L    0                 ; iv_Code
006C          DC.L    0                 ; iv_Node
0070
```

```
0070 ;--------------------------------   FLAG InterruptVektor
0070
0070          DC.L    1EDE                ; iv_Data -> disk.resource
0074          DC.L    FC4AB0              ; iv_Code -> IndexInterruptServer
0078          DC.L    1F58                ; iv_Node -> Interrupt-Struktur
007C
007C ************************************************************************
```

Cia.resource Routinen

```
450C ***************  Cia.resource-Modul  ***********************************
450C
450C FC450C  DC.W    4AFC                ; rt_Matchword
450E         DC.L    FC450C              ; rt_Matchtag
4512         DC.L    FC4790              ; rt_Endskip
4516         DC.B    1                   ; rt_Flags
4517         DC.B    21                  ; rt_Version
4518         DC.B    8                   ; rt_Type
4519         DC.B    50                  ; rt_Pri
451A         DC.L    FC4526              ; rt_Name
451E         DC.L    FC4550              ; rt_idString
4522         DC.L    FC4578              ; rt_Init
4526
4526 FC4526  DC.B    'cia.resource',0,0
4534
4534 FC4534  DC.B    'ciaa.resource',0
4542
4542 FC4542  DC.B    'ciab.resource',0
4550
4550 FC4550  DC.B    'cia 33.22 (29 Mar 1986)',0D,0A,0,0,0
456C
456C ;--------------------------------   Vektoren für Sprungliste
456C
456C FC456C  DC.W    -1                  ; Relativadressen folgen
456E         DC.W    0124                ; FC4690 -> AddICRVector
4570         DC.W    017A                ; FC46E6 -> RemICRVector
4572         DC.W    019E                ; FC470A -> AbleICR
4574         DC.W    01BC                ; FC4728 -> SetICR
4576         DC.W    -1
4578
```

```
4578 ;--------------------------------   Initialisierung
4578
4578 FC4578  move.l  a2,-(a7)            ; a2 retten
457A         bsr.s   FC45E2              ; ---> Resource-Struktur erzeugen
457C         move.l  #BFE001,22(a2)      ; CIAA-Basisadresse eintragen
4584         move.w  #8,26(a2)           ; cr_IntMask für Interrupt 3
458A         move.l  #FC461C,12(a1)      ; is_Code eintragen
4592         moveq   #3,d0               ; d0 := 3 = Ext Interrupt Ebene 2
4594         jsr     -A8(a6)             ; ---> AddIntServer
4598         lea     -66(pc),a0          ; a0 := FC4534 -> 'ciaa.resource'
459C         bsr.s   FC45D2              ; ---> Resource in System einfügen
459E         bsr.s   FC45E2              ; ---> Resource-Struktur erzeugen
45A0         lea     BFD000,a0           ; a0 := CIAB-Basisadresse
45A6         move.l  a0,22(a2)           ; cr_HWAddr in Struktur eintragen
45AA         moveq   #%11000000,d0       ; RTS und DTR Bits setzen
45AC         or.b    d0,200(a0)          ; CIAB DDRA: RTS/DTR = Output
45B0         or.b    d0,0(a0)            ; CIAB PRA: Bits setzen (Aktiv Lo)
45B4         move.w  #2000,26(a2)        ; cr_IntMask für Interrupt 13
45BA         move.l  #FC4610,12(a1)      ; is_Code eintragen
45C2         moveq   #D,d0               ; d0 := 13 = Ext Interrupt Ebene 6
45C4         jsr     -A8(a6)             ; ---> AddIntServer
45C8         lea     -88(pc),a0          ; a0 := FC4542 -> 'ciab.resource'
45CC         bsr.s   FC45D2              ; ---> Resource in System einfügen
45CE         movea.l (a7)+,a2            ; a2 wiederherstellen
45D0         rts
45D2
45D2 ;-----  Resource in System einfügen
45D2
45D2 FC45D2  move.l  a0,A(a1)            ; Namenszeiger in Interrupt-Knoten
45D6         move.l  a0,A(a2)            ; und in Resource-Knoten eintragen
45DA         movea.l a2,a1               ; a1 := a2 -> Resource-Knoten
45DC         jsr     -1E6(a6)            ; ---> AddResource
45E0         rts
45E2
45E2 ;-----  Resource-Struktur erzeugen
45E2
45E2 FC45E2  lea     -78(pc),a0          ; a0 := FC456C -> Vektortabelle
45E6         suba.l  a1,a1               ; a1 := 0: Keine InitStruct-Daten
45E8         suba.l  a2,a2               ; a2 := 0: Keine Init-Routine
45EA         moveq   #7C,d0              ; d0 := 124 Bytes Datenbereich
```

```
45EC          jsr     -54(a6)             ; ---> MakeLibrary
45F0          movea.l d0,a2               ; d0 := LibraryBase
45F2          move.b  #8,8(a2)            ; Typ 8 (Resource) eintragen
45F8          clr.b   9(a2)               ; Pri := 0
45FC          lea     2A(a2),a1           ; a1 -> cr_IntNode
4600          move.b  #2,8(a1)            ; Typ 2 (Interrupt) eintragen
4606          clr.b   9(a1)               ; Pri := 0
460A          move.l  a2,E(a1)            ; is_Data := LibraryBase
460E          rts
4610
4610 ;--------------------------------   CIAB InterruptServer
4610
4610 FC4610   movem.l d2/a2,-(a7)         ; Register retten
4614          move.b  BFDD00,d2           ; d2 := CIAB ICR
461A          bra.s   FC4626              ; --->
461C
461C ;--------------------------------   CIAA InterruptServer
461C
461C FC461C   movem.l d2/a2,-(a7)         ; Register retten
4620          move.b  BFED01,d2           ; d2 := CIAA ICR
4626 FC4626   bclr.l  #7,d2               ; SET-Bit löschen
462A          or.b    29(a1),d2           ; cr_IActive-Bits setzen
462E          move.b  d2,29(a1)           ; und neu speichern
4632          and.b   28(a1),d2           ; cr_IEnable
4636          beq.s   FC4658              ; Kein Interrupt freigegeben: ->
4638          movea.l a1,a2               ; a2 := a1 -> CIA-Resource
463A          eor.b   d2,29(a2)           ; Bits in cr_IActive flippen
463E          lsr.b   #1,d2               ; Bit 0 herausschieben
4640          bcs.s   FC465E              ; TimerA-Interrupt: ->
4642 FC4642   lsr.b   #1,d2               ; Bit 1 herausschieben
4644          bcs.s   FC4668              ; TimerB-Interrupt: ->
4646          beq.s   FC4658              ; Kein Flag mehr gesetzt: ->
4648 FC4648   lsr.b   #1,d2               ; Bit 2 herausschieben
464A          bcs.s   FC4672              ; Alarm-Interrupt: ->
464C          beq.s   FC4658              ; Kein Flag mehr gesetzt: ->
464E FC464E   lsr.b   #1,d2               ; Bit 3 herausschieben
4650          bcs.s   FC467C              ; SerialPort-Interrupt: ->
4652          beq.s   FC4658              ; Kein Flag mehr gesetzt: ->
4654 FC4654   lsr.b   #1,d2               ; Bit 4 herausschieben
4656          bcs.s   FC4686              ; FLG-Interrupt: ->
```

```
4658 FC4658  movem.l (a7)+,d2/a2          ; Register wiederherstellen
465C         rts
465E
465E FC465E  movem.l 40(a2),a1/a5         ; TimerA-Interrupt
4664         jsr     (a5)                 ; ---> Interrupt bearbeiten
4666         bra.s   FC4642               ; ---> zurück in die Abfrage
4668
4668 FC4668  movem.l 4C(a2),a1/a5         ; TimerB-Interrupt
466E         jsr     (a5)                 ; ---> Interrupt bearbeiten
4670         bra.s   FC4648               ; ---> zurück in die Abfrage
4672
4672 FC4672  movem.l 58(a2),a1/a5         ; Alarm-Interrupt
4678         jsr     (a5)                 ; ---> Interrupt bearbeiten
467A         bra.s   FC464E               ; ---> zurück in die Abfrage
467C
467C FC467C  movem.l 64(a2),a1/a5         ; SerialPort-Interrupt
4682         jsr     (a5)                 ; ---> Interrupt bearbeiten
4684         bra.s   FC4654               ; ---> zurück in die Abfrage
4686
4686 FC4686  movem.l 70(a2),a1/a5         ; FLG-Interrupt
468C         jsr     (a5)                 ; ---> Interrupt bearbeiten
468E         bra.s   FC4658               ; ---> zurück in die Abfrage
4690
4690 ;--------------------------------   AddICRVector
4690
4690 FC4690  moveq   #0,d1                ; d1 := 0
4692         move.b  d0,d1                ; d1 := Interrupt-Nummer
4694         mulu    #C,d1                ; d1 := Offset für IntVektor
4698         movea.l 4,a0                 ; a0 := SysBase
469C         move.w  #4000,INTENA         ; Interrupts abschalten
46A4         addq.b  #1,126(a0)           ; Disable
46A8         lea     40(a6,d1.w),a0       ; a0 -> IntVektor in CIA-Resource
46AC         move.l  8(a0),d1             ; d1 := iv_Node
46B0         bne.s   FC46E2               ; Eintrag vorhanden: ->
46B2         move.l  a1,8(a0)             ; iv_Node := a1 -> Interrupt-Struktur
46B6         move.l  12(a1),4(a0)         ; iv_Code := is_Code
46BC         move.l  E(a1),0(a0)          ; iv_Data := is_Data
46C2         move.w  #80,d1               ; SET-Bit setzen
46C6 FC46C6  bset.l  d0,d1                ; Interrupt-Bit (d0) setzen
46C8         move.w  d1,d0                ; d0 := Bitmuster für ICR
```

```
46CA          bsr.s   FC470A              ; ---> AbleICR
46CC          moveq   #0,d0               ; d0 := 0
46CE FC46CE   movea.l 4,a0                ; a0 := SysBase
46D2          subq.b  #1,126(a0)          ; Enable
46D6          bge.s   FC46E0              ; Noch keine Freigabe: ->
46D8          move.w  #C000,INTENA        ; Interrupts freigeben
46E0 FC46E0   rts
46E2
46E2 FC46E2   move.l  d1,d0               ; d0 := d1 = iv_Node
46E4          bra.s   FC46CE              ; --->
46E6
46E6 ;--------------------------------   RemICRVector
46E6
46E6 FC46E6   moveq   #0,d1               ; d1 := 0
46E8          move.b  d0,d1               ; d1 := Interrupt Nummer
46EA          mulu    #C,d1               ; d1 := Offset für IntVektor
46EE          movea.l 4,a0                ; a0 := SysBase
46F2          move.w  #4000,INTENA        ; Interrupts sperren
46FA          addq.b  #1,126(a0)          ; Disable
46FE          lea     40(a6,d1.w),a0      ; a0 -> IntVektor in CIA-Resource
4702          clr.l   8(a0)               ; iv_Node löschen
4706          moveq   #0,d1               ; d1 := 0
4708          bra.s   FC46C6              ; --->
470A
470A ;--------------------------------   AbleICR
470A
470A FC470A   movea.l 4,a0                ; a0 := SysBase
470E          move.w  #4000,INTENA        ; Interrupts sperren
4716          addq.b  #1,126(a0)          ; Disable
471A          movea.l 22(a6),a0           ; a0 -> cr_HWAddr = CIA-Basisadresse
471E          move.b  d0,D00(a0)          ; d0 in ICR schreiben
4722          lea     28(a6),a1           ; a1 -> cr_IEnable
4726          bra.s   FC474C              ; --->
4728
4728 ;--------------------------------   SetICR
4728
4728 FC4728   movea.l 4,a0                ; a0 := SysBase
472C          move.w  #4000,INTENA        ; Interrupts sperren
4734          addq.b  #1,126(a0)          ; Disable
4738          movea.l 22(a6),a0           ; a0 -> cr_HWAddr = CIA-Basisadresse
```

```
473C          move.b  D00(a0),d1          ; d0 aus ICR lesen
474Ø          bclr.l  #7,d1               ; Bit 7 löschen
4744          or.b    d1,29(a6)           ; in cr_IActive einodern
4748          lea     29(a6),a1           ; a1 -> cr_IActive
474C FC474C   moveq   #0,d1               ; d1 := 0
474E          move.b  (a1),d1             ; d1 := cr_IActive bzw. cr_IEnable
4750          tst.b   d0                  ; ICR-Flags gesetzt?
4752          beq.s   FC4762              ; Nein: ->
4754          bclr.l  #7,d0               ; SET-Bit löschen
4758          bne.s   FC476Ø              ; Anderes Bit gesetzt: ->
475A          not.b   d0                  ; Alle Bits flippen
475C          and.b   d0,(a1)             ; AND mit cr_IActive/cr_IEnable
475E          bra.s   FC4762              ; --->
476Ø
476Ø FC476Ø   or.b    d0,(a1)             ; OR mit cr_IActive/cr_IEnable
4762 FC4762   move.b  28(a6),d0           ; d0 := cr_IEnable
4766          and.b   29(a6),d0           ; d0 mit cr_IActive maskieren
476A          beq.s   FC477A              ; Ergebnis 0: ->
476C          move.w  26(a6),d0           ; d0 := cr_IntMask
477Ø          ori.w   #8000,d0            ; Bit 15 setzen
4774          move.w  d0,DFF09C           ; Bits in INTREQ setzen
477A FC477A   movea.l 4,a0                ; a0 := SysBase
477E          subq.b  #1,126(a0)          ; Enable
4782          bge.s   FC478C              ; Noch keine Freigabe: ->
4784          move.w  #C000,INTENA        ; Interrupts freigeben
478C FC478C   move.l  d1,d0               ; d0 := d1 = cr_IActive bzw. cr_IEnable
478E          rts
4790
4790 ************************************************************************
```

Potgo.resource Datenbereich

```
-Ø12 *************** potgo.resource ***************************************
-Ø12
-Ø12          jmp     FE4996              ; ---> WritePotgo
-ØØC          jmp     FE498Ø              ; ---> FreePotBits
-ØØ6          jmp     FE4916              ; ---> AllocPotBits
ØØØØ
ØØØØ ;-------------------------------  Library-Block
ØØØØ
ØØØØ          DC.L    1A18                ; ln_Succ
ØØØ4          DC.L    Ø7C6                ; ln_Pred
ØØØ8          DC.B    8                   ; ln_Type (8 = Resource)
ØØØ9          DC.B    Ø                   ; ln_Pri
ØØØA          DC.L    FE489A              ; ln_Name -> 'potgo.resource'
ØØØE
ØØØE          DC.B    6                   ; lib_Flags
ØØØF          DC.B    Ø                   ; lib_Pad
ØØ1Ø          DC.W    12                  ; lib_NegSize
ØØ12          DC.W    29                  ; lib_PosSize
ØØ14          DC.W    Ø                   ; lib_Version
ØØ16          DC.W    Ø                   ; lib_Revision
ØØ18          DC.L    Ø                   ; lib_IdString
ØØ1C          DC.L    Ø                   ; lib_LibSum
ØØ2Ø          DC.W    Ø                   ; lib_OpenCnt
ØØ22
ØØ22 ;-------------------------------   Verschiedene Parameter
ØØ22
ØØ22          DC.L    676                 ; SysBase
ØØ26
ØØ26          DC.W    ØFØØ                ; Allozierte PotBits
ØØ28          DC.W    ØFØØ                ; Aktuelle PotBits
ØØ2A
ØØ2A ************************************************************************
```

Potgo.resource Routinen

```
488Ø ********** potgo.resource-Modul **********************************
488Ø
488Ø FE488Ø  DC.W    4AFC                ; rt_Matchword
4882         DC.L    FE488Ø              ; rt_Matchtag
4886         DC.L    FE49C8              ; rt_EndSkip
488A         DC.B    1                   ; rt_Flags
488B         DC.B    21                  ; rt_Version
488C         DC.B    8                   ; rt_Type (8 = Resource)
488D         DC.B    64                  ; rt_Pri = 1ØØ
488E         DC.L    FE489A              ; rt_Name
4892         DC.L    FE48AA              ; rt_IdString
4896         DC.L    FE48C8              ; rt_Init
489A
489A FE489A  DC.B    'potgo.resource',Ø,Ø
48AA
48AA FE48AA  DC.B    'potgo 33.7 (1Ø Feb 1986)',ØD,ØA,Ø,Ø,Ø,Ø
48C8
48C8 ;--------------------------------   Initialisierung
48C8
48C8 FE48C8  move.l  a2,-(a7)            ; a2 retten
48CA         lea     4Ø(pc),aØ           ; aØ := FE49ØC -> Vektortabelle
48CE         lea     26(pc),a1           ; a1 := FE48F6 -> InitStruct-Tabelle
48D2         lea     1A(pc),a2           ; a2 := FE48EE -> Init-Routine
48D6         move.l  #29,dØ              ; dØ := 41 = Bytezahl für Library
48DC         jsr     -54(a6)             ; ---> MakeLibrary
48EØ         tst.l   dØ                  ; Library ok?
48E2         beq.s   FE48EA              ; nein: ->
48E4         movea.l dØ,a1               ; a1 := dØ -> potgo.resource
48E6         jsr     -1E6(a6)            ; ---> AddResource
48EA FE48EA  movea.l (a7)+,a2            ; a2 wiederherstellen
48EC         rts
48EE
```

```
48EE ;--------------------------------   Init-Routine für MakeLibrary
48EE
48EE FE48EE  movea.l d0,a0               ; a0 := d0 -> potgo.resource
48F0         move.l  a6,22(a0)           ; SysBase speichern
48F4         rts
48F6
48F6 ;--------------------------------   InitStruct-Tabelle
48F6
48F6 FE48F6  DC.W    E000               ; InitByte
48F8         DC.W    8                  ; ln_Type
48FA         DC.B    8                  ; = nt_Resource
48FB         DC.B    0
48FC         DC.W    C000               ; InitLong
48FE         DC.W    A                  ; ln_Name
4900         DC.L    FE489A             ; -> 'potgo.resource'
4904         DC.W    E000               ; InitByte
4906         DC.W    E                  ; lib_Flags
4908         DC.B    6                  ; = Changed, Sumused
4909         DC.B    0
490A         DC.W    0
490C
490C ;--------------------------------   Vektortabelle für Sprungliste
490C
490C FE490C  DC.W    -1                 ; Relativadressen folgen
490E         DC.W    000A               ; FE4916 -> AllocPotBits
4910         DC.W    0074               ; FE4980 -> FreePotBits
4912         DC.W    008A               ; FE4996 -> WritePotgo
4914         DC.W    -1                 ; Endemarke
4916
4916 ;--------------------------------   AllocPotBits
4916
4916 FE4916  movem.l d2,-(a7)           ; d2 retten
491A         move.l  a6,-(a7)           ; a6 retten
491C         movea.l 22(a6),a6          ; a6 := SysBase
4920         jsr     -78(a6)            ; ---> Disable
4924         movea.l (a7)+,a6           ; a6 wiederherstellen
4926         move.w  26(a6),d2          ; d2 := zugeordnete PotBits
492A         move.w  d2,d1              ; d1 := d2
492C         andi.w  #5500,d1           ; DAT-Bits isolieren
4930         lsl.w   #1,d1              ; in OUT-Bits verschieben
```

```
4932          or.w    d2,d1                ; zugeordnete PotBits einodern
4934          not.w   d1                   ; Ergebnis invertieren
4936          and.w   d1,d0                ; Zugeordnete Bits löschen
4938          btst.l  #0,d0                ; Startbit angefordert?
493C          beq.s   FE4950               ; Nein: ->
493E          move.w  d2,d1                ; d1 := d2 = zugeordnete PotBits
4940          lsr.w   #1,d1                ; OUT-Bits in DAT-Bits verschieben
4942          not.w   d1                   ; Ergebnis invertiern
4944          andi.w  #5500,d1             ; DAT-Bits isolieren
4948          and.w   d2,d1                ; Nicht zugeordnete PotBits löschen
494A          beq.s   FE4950               ; Ergebnis = 0: ->
494C          bclr.l  #0,d0                ; Startbit löschen
4950 FE4950   btst.l  #0,d2                ; Startbit zugeordnet?
4954          beq.s   FE4966               ; Nein: ->
4956          move.w  d0,d1                ; d1 := d0 = PotBit-Anforderung
4958          lsr.w   #1,d1                ; OUT-Bits in DAT-Bits verschieben
495A          andi.w  #5500,d0             ; DAT-Bits isolieren
495E          and.w   d1,d0                ; Nicht angeforderte Bits löschen
4960          move.w  d0,d1                ; d1 := d0 = Ergebnis
4962          lsl.w   #1,d1                ; DAT-Bits in OUT-Bits zurückschieben
4964          or.w    d1,d0                ; DAT-Bits wieder einodern
4966 FE4966   andi.w  #FF01,d0             ; Reservebits löschen
496A          or.w    d0,26(a6)            ; Zugeordnete Bits := Ergebnis
496E          move.l  a6,-(a7)             ; a6 retten
4970          movea.l 22(a6),a6            ; a6 := SysBase
4974          jsr     -7E(a6)              ; ---> Enable
4978          movea.l (a7)+,a6             ; a5 wiederherstellen
497A          movem.l (a7)+,d2             ; d2 wiederherstellen
497E          rts
4980
4980 ;--------------------------------   FreePotBits
4980
4980 FE4980   andi.w  #5501,d0             ; Databits und Startbit isolieren
4984          move.w  d0,d1                ; d1 := d0 = Databits
4986          lsl.w   #1,d1                ; um 1 Bit linksverschieben
4988          andi.w  #AA00,d1             ; Verschobenes Startbit löschen
498C          or.w    d1,d0                ; Output-Bits einodern
498E          not.w   d0                   ; Ergebnis invertieren
```

```
4990          and.w   d0,26(a6)           ; Data- und Output-Bits löschen
4994          rts
4996
4996 ;--------------------------------    WritePotgo
4996
4996 FE4996   and.w   d1,d0               ; d0 := Bits maskiert
4998          not.w   d1                  ; Maske invertieren
499A          move.l  a6,-(a7)            ; a6 retten
499C          movea.l 22(a6),a6           ; a6 := SysBase
49A0          jsr     -78(a6)             ; ---> Disable
49A4          movea.l (a7)+,a6            ; a6 wiederherstellen
49A6          and.w   d1,28(a6)           ; Maskenbits in aktuellen PotBits := 0
49AA          or.w    28(a6),d0           ; Übrige Bits einodern
49AE          move.w  d0,DFF034           ; POTGO := d0
49B4          clr.b   d0                  ; Start- und reservierte Bits löschen
49B6          move.w  d0,28(a6)           ; Aktuelle Bits speichern
49BA          move.l  a6,-(a7)            ; a6 retten
49BC          movea.l 22(a6),a6           ; a6 := SysBase
49C0          jsr     -7E(a6)             ; ---> Enable
49C4          movea.l (a7)+,a6            ; a6 wiederherstellen
49C6          rts
49C8
49C8 ***************************************************************************
```

Disk.resource Datenbereich

```
-01E ***************  disk.resource  ************************************
-01E
-01E          jmp     FC4A74              ; ---> dr_GetUnitID
-018          jmp     FC4A0E              ; ---> dr_GiveUnit
-012          jmp     FC4996              ; ---> dr_GetUnit
-00C          jmp     FC4A6E              ; ---> dr_FreeUnit
-006          jmp     FC4A62              ; ---> dr_AllocUnit
0000
0000 ;--------------------------------   Library-Block
0000
0000          DC.L    1BA4                ; ln_Succ
0004          DC.L    1B18                ; ln_Pred
0008          DC.B    8                   ; ln_Type (8 = Resource)
0009          DC.B    0                   ; ln_Pri
000A          DC.L    FC47AE              ; ln_Name -> 'disk.resource'
000E
000E          DC.B    6                   ; lib_Flags = Changed, Sumused
000F          DC.B    0                   ; lib_Pad
0010          DC.W    1E                  ; lib_NegSize
0012          DC.W    90                  ; lib_PosSize
0014          DC.W    21                  ; lib_Version
0016          DC.W    12                  ; lib_Revision
0018          DC.L    0                   ; lib_IdString
001C          DC.L    0                   ; lib_Sum
002O          DC.W    0                   ; lib_OpenCnt
0022
0022 ;--------------------------------   Verschiedene Parameter
0022
0022          DC.L    6108                ; dr_Current -> aktuelle Unit-Struktur
0026          DC.B    3                   ; dr_Flags (Alloc0-Alloc3, Active 7)
0027          DC.B    0                   ; dr_Pad
0028          DC.L    676                 ; dr_SysLib
```

```
002C        DC.L    1B18            ; dr_ciaResource
0030
0030        DC.L    0               ; dr_UnitID 0
0034        DC.L    0               ;           1
0038        DC.L    -1              ;           2
003C        DC.L    -1              ;           3
0040
0040 ;--------------------------------  Warteliste dr_Waiting
0040
0040        DC.L    1F22            ; lh_Head
0044        DC.L    0               ; lh_Tail
0048        DC.L    1F1E            ; lh_TailPred
004C        DC.B    0               ; lh_Type
004D        DC.B    0               ; lh_Pad
004E
004E ;--------------------------------  Interrupt-Struktur dr_DiskBlock
004E
004E        DC.L    0               ; ln_Succ
0052        DC.L    0               ; ln_Pred
0056        DC.B    0               ; ln_Type
0057        DC.B    0               ; ln_Pri
0058        DC.L    0               ; ln_Name
005C        DC.L    *0000           ; is_Data
0060        DC.L    FC4A80          ; is_Code
0064
0064 ;--------------------------------  Interrupt-Struktur dr_DiskSync
0064
0064        DC.L    0               ; ln_Succ
0068        DC.L    0               ; ln_Pred
006C        DC.B    0               ; ln_Type
006D        DC.B    0               ; ln_Pri
006E        DC.L    0               ; ln_Name
0072        DC.L    *0000           ; is_Data
0076        DC.L    FC4A98          ; is_Code
007A
007A ;--------------------------------  Interrupt-Struktur dr_Index
007A
007A        DC.L    0               ; ln_Succ
007E        DC.L    0               ; ln_Pred
0082        DC.B    0               ; ln_Type
```

```
0083          DC.B    0                   ; ln_Pri
0084          DC.L    0                   ; ln_Name
0088          DC.L    *0000               ; is_Data
008C          DC.L    FC4AB0              ; is_Code
0090
0090 ************************************************************************
```

Disk.resource Routinen

```
4794 ***************  Disk.resource-Modul  ********************************
4794
4794 FC4794  DC.W    4AFC                ; rt_Matchword
4796         DC.L    FC4794              ; rt_Matchtag
479A         DC.L    FC4AF4              ; rt_EndSkip
479E         DC.B    1                   ; rt_Flags
479F         DC.B    21                  ; rt_Version
47AØ         DC.B    8                   ; rt_Type (8 = Resource)
47A1         DC.B    46                  ; rt_Pri = 7Ø
47A2         DC.L    FC47AE              ; rt_Name
47A6         DC.L    FC47BC              ; rt_IdString
47AA         DC.L    FC47D8              ; rt_Init
47AE
47AE FC47AE  DC.B    'disk.resource',Ø
47BC
47BC FC47BC  DC.B    'disk 33.18 (6 May 1986)',ØD,ØA,Ø,Ø,Ø
47D8
47D8 ;--------------------------------   Initialisierung
47D8
47D8 FC47D8  movem.l d2-d3/a2-a3/a6,-(a7) ; Register retten
47DC         movea.l 4,a6                ; a6 := SysBase
47E2         lea     FC495C,a1           ; a1 -> 'ciab.resource'
47E8         jsr     -1F2(a6)            ; ---> OpenResource
47EC         move.l  dØ,d2               ; d2 := dØ -> ciab.resource oder = Ø
47EE         bne.s   FC48ØA              ; Resource geöffnet: ->
47FØ         movem.l d7/a5-a6,-(a7)      ; Register retten
47F4         move.l  #21Ø58Ø2Ø,d7        ; d7 := AlertKode
47FA         movea.l 4,a6                ; a6 := SysBase
47FE         jsr     -6C(a6)             ; ---> Alert
48Ø2         movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
48Ø6         bra     FC4916              ; ---> Abschluß
48ØA
```

```
480A FC480A  lea     FC4ADA,a0          ; a0 -> Vektortabelle
4810         lea     FC496A,a1          ; a1 -> InitStruct-Tabelle
4816         suba.l  a2,a2              ; a2 := 0: Keine Init-Routine
4818         move.l  #90,d0             ; d0 := 144 Bytes für Library
481E         jsr     -54(a6)            ; ---> MakeLibrary
4822         tst.l   d0                 ; Library ok?
4824         bne.s   FC4840             ; ja: ->
4826         movem.l d7/a5-a6,-(a7)     ; Register retten
482A         move.l  #21020000,d7       ; d7 := AlertKode
4830         movea.l 4,a6               ; a6 := SysBase
4834         jsr     -6C(a6)            ; ---> Alert
4838         movem.l (a7)+,d7/a5-a6     ; Register wiederherstellen
483C         bra     FC4916             ; ---> Abschluß
4840
4840 FC4840  movea.l d0,a2              ; a2 := d0 -> disk.resource
4842         lea     4E(a2),a1          ; a1 -> Int-Struktur dr_DiskBlock
4846         move.l  #FC4A80,12(a1)     ; is_Code eintragen
484E         move.l  a2,E(a1)           ; is_Data eintragen
4852         moveq   #1,d0              ; d0 := Interrupt-Nummer
4854         jsr     -A2(a6)            ; ---> SetIntVector
4858         lea     64(a2),a1          ; a1 -> Int-Struktur dr_DiskSync
485C         move.l  #FC4A98,12(a1)     ; is_Code eintragen
4864         move.l  a2,E(a1)           ; is_Data eintragen
4868         moveq   #C,d0              ; d0 := Interrupt-Nummer
486A         jsr     -A2(a6)            ; ---> SetIntVector
486E         lea     BFD000,a1          ; a1 := ciab-Base
4874         move.b  #FF,100(a1)        ; PRB Alle Ausgänge := 1
487A         move.b  #FF,300(a1)        ; DDRB Alle Bits := Ausgang
4880         andi.b  #C3,BFE201         ; DDRA Bits 2,3,4,5 := Eingang
4888         move.w  #4000,DFF09A       ; Interrupts sperren
4890         addq.b  #1,126(a6)         ; IDNestCnt erhöhen
4894         move.l  d2,2C(a2)          ; ciab.resource eintragen
4898         lea     7A(a2),a1          ; a1 -> Int-Struktur dr_Index
489C         move.l  a2,E(a1)           ; is_Data eintragen
48A0         move.l  #FC4AB0,12(a1)     ; is_Code eintragen
48A8         moveq   #4,d0              ; d0 := Interrupt-Nummer FLG-Interrupt
48AA         move.l  a6,-(a7)           ; a6 retten
48AC         movea.l d2,a6              ; a6 -> ciab.resource
48AE         jsr     -6(a6)             ; ---> AddICRVector
48B2         movea.l (a7)+,a6           ; a6 wiederherstellen
```

```
48B4          moveq   #1Ø,dØ              ; dØ := FLG-Interrupt-Bit im ICR
48B6          move.l  a6,-(a7)            ; a6 retten
48B8          movea.l d2,a6               ; a6 -> ciab.resource
48BA          jsr     -12(a6)             ; ---> AbleICR
48BE          movea.l (a7)+,a6            ; a6 wiederherstellen
48CØ          subq.b  #1,126(a6)          ; IDNestCnt erniedrigen
48C4          bge.s   FC48CE              ; Interrupts noch gesperrt: ->
48C6          move.w  #CØØØ,DFFØ9A        ; Interrupts freigeben
48CE FC48CE   lea     DFFØØØ,aØ           ; aØ -> ChipRegister Base
48D4          move.l  a6,28(a2)           ; SysBase in disk.resource eintragen
48D8          move.w  #4ØØØ,24(aØ)        ; DSKLEN := 16384
48DE          move.w  #8Ø1Ø,96(aØ)        ; DMACON := SET & DSKEN
48E4          lea     4Ø(a2),aØ           ; aØ -> Warteliste dr_Waiting
48E8          move.l  aØ,(aØ)             ; Macro NewList
48EA          addq.l  #4,(aØ)             ; :
48EC          clr.l   4(aØ)               ; :
48FØ          move.l  aØ,8(aØ)            ; :
48F4          move.l  #Ø,3Ø(a2)           ; dr_UnitID Ø := drt_Amiga
48FC          moveq   #2,d2               ; d2 := 2
48FE          move.b  #1Ø,d3              ; d3 := SEL1 (Select external drive 1)
49Ø2          lea     34(a2),a3           ; a3 -> dr_UnitID 1
49Ø6 FC49Ø6   bsr     FC491C              ; ---> Externes Laufwerk suchen
49ØA          lsl.b   #1,d3               ; Nächstes Select-Bit setzen
49ØC          dbra    d2,FC49Ø6           ; und wiederholen, bis d2 = -1 ->
491Ø          movea.l a2,a1               ; a1 := a2 -> disk.resource
4912          jsr     -1E6(a6)            ; ---> AddResource
4916 FC4916   movem.l (a7)+,d2-d3/a2-a3/a6 ; Register wiederherstellen
491A          rts
491C
491C ;------ Externes Laufwerk suchen
491C
491C FC491C   not.b   d3                  ; Select-Bits invertieren
491E          lea     BFD1ØØ,aØ           ; aØ -> ciab.PRB
4924          move.b  #7F,dØ              ; Alle Laufwerke deselektieren
4928          move.b  dØ,(aØ)
492A          and.b   d3,dØ               ; Select-Bit löschen
492C          move.b  dØ,(aØ)             ; Laufwerk selektieren, Motor ein
492E          move.b  #FF,(aØ)            ; Laufwerk deselektieren
4932          move.b  d3,(aØ)             ; Laufwerk selektieren, Motor aus
4934          move.b  #FF,(aØ)            ; Laufwerk deselektieren
```

```
4938          moveq   #1F,d1              ; d1 := 31 = Zähler für 32 Bits
493A          moveq   #0,d0               ; d0 := 0
493C FC493C   lsl.l   #1,d0               ; d0 um 1 Bit links verschieben
493E          move.b  d3,(a0)             ; Laufwerk wählen
4940          btst.b  #5,BFE001           ; RDY (Disk ready) testen
4948          beq.s   FC494E              ; RDY = 0: Laufwerk vorhanden ->
494A          bset.l  #0,d0               ; sonst Bit 0 in d0 setzen
494E FC494E   move.b  #FF,(a0)            ; Motor ausschalten
4952          dbra    d1,FC493C           ; 31 mal wiederholen ->
4956          move.l  d0,(a3)+            ; Laufwerktyp eintragen (0,55555555,-1)
4958          not.b   d3                  ; d3 wiederherstellen
495A          rts
495C
495C FC495C   DC.B    'ciab.resource',0
496A
496A ;--------------------------------   Tabelle für InitStruct
496A
496A FC496A   DC.W    E000                ; InitByte
496C          DC.W    8                   ; ln_Type
496E          DC.B    8                   ; = 8 (nt_Resource)
496F          DC.B    0
4970          DC.W    C000                ; InitLong
4972          DC.W    A                   ; ln_Name
4974          DC.L    FC47AE              ; -> 'disk.resource'
4978          DC.W    E000                ; InitByte
497A          DC.W    E                   ; lib_Flags
497C          DC.B    6                   ; = Changed, Sumused
497D          DC.B    0
497E          DC.W    D000                ; InitWord
4980          DC.W    14                  ; lib_Version
4982          DC.W    21                  ; = 33
4984          DC.W    D000                ; InitWord
4986          DC.W    16                  ; lib_Revision
4988          DC.W    12                  ; = 18
498A          DC.W    C000                ; InitLong
498C          DC.W    22                  ; dr_Current
498E          DC.L    -1                  ; löschen
4992          DC.W    0
4994
```

```
4994          DC.W    0
4996
4996 ;--------------------------------   GetUnit
4996
4996 FC4996   movea.l 4,a0                ; a0 := SysBase
499A          move.w  #4000,DFF09A        ; Interrupts sperren
49A2          addq.b  #1,126(a0)          ; IDNestCnt erhöhen
49A6          bset.b  #7,26(a6)           ; Active-Bit setzen
49AC          bne.s   FC49BA              ; War schon gesetzt: ->
49AE          move.l  22(a6),d0           ; d0 := dr_Current
49B2          move.l  a1,22(a6)           ; dr_Current := a1
49B6          bra     FC49FA              ; ---> Abschluß
49BA
49BA FC49BA   cmpa.l  22(a6),a1           ; gleiche Unit wie angefordert?
49BE          bne.s   FC49DE              ; nein: ->
49C0          move.l  a1,-(a7)            ; a1 retten
49C2          movea.l a7,a1               ; a1 := a7 -> Zeiger auf Unit-Struktur
49C4          movem.l d7/a5-a6,-(a7)      ; Register retten
49C8          move.l  #21000001,d7        ; d7 := AlertKode = AN_DRHasDisk
49CE          lea     (a1),a5             ; a5 -> Unit-Struktur
49D0          movea.l 4,a6                ; a6 := SysBase
49D4          jsr     -6C(a6)             ; ---> Alert
49D8          movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
49DC          movea.l (a7)+,a1            ; a1 wiederherstellen
49DE FC49DE   lea     40(a6),a0           ; a0 -> Header der Warteliste
49E2          lea     4(a0),a0            ; Macro AddTail
49E6          move.l  4(a0),d0            ; :
49EA          move.l  a1,4(a0)            ; :
49EE          move.l  a0,(a1)             ; :
49F0          move.l  d0,4(a1)            ; :
49F4          movea.l d0,a0               ; :
49F6          move.l  a1,(a0)             ; :
49F8          moveq   #0,d0               ; d0 := 0
49FA FC49FA   movea.l 4,a0                ; a0 := SysBase
49FE          subq.b  #1,126(a0)          ; IDNestCnt erniedrigen
4A02          bge.s   FC4A0C              ; Interrupts noch gesperrt: ->
4A04          move.w  #C000,DFF09A        ; Interrupts freigeben
4A0C FC4A0C   rts
4A0E
```

```
4A0E ;--------------------------------   GiveUnit
4A0E
4A0E FC4A0E   movea.l 4,a0               ; a0 := SysBase
4A12          move.w  #4000,DFF09A       ; Interrupts sperren
4A1A          addq.b  #1,126(a0)         ; IDNestCnt erhöhen
4A1E          move.b  #FF,BFD100         ; Alle Laufwerke deselektieren
4A26          bclr.b  #7,26(a6)          ; Active-Bit löschen
4A2C          lea     40(a6),a0          ; a0 -> Wartelisten-Header
4A30          move.l  a6,-(a7)           ; a6 retten
4A32          movea.l 28(a6),a6          ; a6 := SysBase
4A36          jsr     -102(a6)           ; ---> RemHead
4A3A          movea.l (a7)+,a6           ; a6 wiederherstellen
4A3C          movea.l 4,a0               ; a0 := SysBase
4A40          subq.b  #1,126(a0)         ; IDNestCnt erniedrigen
4A44          bge.s   FC4A4E             ; Interrupts noch gesperrt: ->
4A46          move.w  #C000,DFF09A       ; Interrupts freigeben
4A4E FC4A4E   tst.l   d0                 ; War Warteliste leer?
4A50          beq.s   FC4A60             ; ja: Fertig ->
4A52          movea.l d0,a1              ; a1 := d0 -> Entfernte Unit-Struktur
4A54          move.l  a6,-(a7)           ; a6 retten
4A56          movea.l 28(a6),a6          ; a6 := SysBase
4A5A          jsr     -17A(a6)           ; ---> ReplyMsg
4A5E          movea.l (a7)+,a6           ; a6 wiederherstellen
4A60 FC4A60   rts
4A62
4A62 ;--------------------------------   AllocUnit
4A62
4A62 FC4A62   move.l  d0,d1              ; d1 := d0 = Unit-Nummer (0,1,2,3)
4A64          moveq   #0,d0              ; d0 := 0 = Fehlerflag
4A66          bset.b  d1,26(a6)          ; Unit-Bit setzen
4A6A          seq     d0                 ; War nicht gesetzt: d0 := -1 = ok-Flag
4A6C          rts
4A6E
4A6E ;--------------------------------   FreeUnit
4A6E
4A6E FC4A6E   bclr.b  d0,26(a6)          ; Unit-Bit zurücksetzen
4A72          rts
4A74
```

```
4A74 ;--------------------------------   GetUnitID
4A74
4A74 FC4A74  lea     30(a6),a0           ; a0 -> dr_UnitID 0
4A78         lsl.l   #2,d0               ; d0 := Offset in Tabelle
4A7A         move.l  0(a0,d0.w),d0       ; d0 := angeforderte UnitID
4A7E         rts
4A80
4A80 ;--------------------------------   InterruptServer
4A80
4A80 ;------ DiskBlock-InterruptServer
4A80
4A80 FC4A80  move.w  #2,DFF09C           ; INTREQ.DSKBLK setzen
4A88         move.l  22(a1),d0           ; d0 := dr_Current
4A8C         beq.s   FC4AC0              ; = 0: Fehler ->
4A8E         movea.l d0,a1               ; a1 := d0 -> aktuelle Unit-Struktur
4A90         movem.l 22(a1),a1/a5        ; dru_DiskBlock-Zeiger übernehmen
4A96         jmp     (a5)                ; ---> InterruptServer
4A98
4A98 ;------ DiskSync-InterruptServer
4A98
4A98 FC4A98  move.w  #1000,DFF09C; INTREQ.DSKSYN setzen
4AA0         move.l  22(a1),d0           ; d0 := dr_Current
4AA4         beq.s   FC4AC0              ; = 0: Fehler ->
4AA6         movea.l d0,a1               ; a1 := d0 -> aktuelle Unit-Struktur
4AA8         movem.l 38(a1),a1/a5        ; dru_DiskSync-Zeiger übernehmen
4AAE         jmp     (a5)                ; ---> InterruptServer
4AB0
4AB0 ;------ Index-InterruptServer
4AB0
4AB0 FC4AB0  move.l  22(a1),d0           ; d0 := dr_Current
4AB4         beq.s   FC4AD8              ; = 0: Fertig ->
4AB6         movea.l d0,a1               ; a1 := d0 -> aktuelle Unit-Struktur
4AB8         movem.l 4E(a1),a1/a5        ; dru_Index-Zeiger übernehmen
4ABE         jmp     (a5)                ; ---> InterruptServer
4AC0
4AC0 ;------ Fehlerausgang
4AC0
4AC0 FC4AC0  movem.l d7/a5-a6,-(a7)      ; Register retten
4AC4         move.l  #21000002,d7        ; d7 := AlertKode AN_DRIntNoAct
4ACA         movea.l 4,a6                ; a6 := SysBase
```

```
4ACE          jsr     -6C(a6)             ; ---> Alert
4AD2          movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
4AD6          rts
4AD8
4AD8 FC4AD8   rts
4ADA
4ADA ;--------------------------------   Vektortabelle für Sprungliste
4ADA
4ADA FC4ADA   DC.L    FC4A62              ; AllocUnit
4ADE          DC.L    FC4A6E              ; FreeUnit
4AE2          DC.L    FC4996              ; GetUnit
4AE6          DC.L    FC4A0E              ; GiveUnit
4AEA          DC.L    FC4A74              ; GetUnitID
4AEE          DC.L    -1
4AF2
4AF2          DC.W    0
4AF4
4AF4 ************************************************************************
```

Misc.resource Datenbereich

```
-00C ************************* misc.resource *****************************
-00C
-00C        jmp     $FE4862         ; ---> FreeMiscResource
-006        jmp     $FE482A         ; ---> GetMiscResource
0000
0000 ;-------------------------   Library-Block
0000
0000        DC.L    07CA            ; ln_Succ
0004        DC.L    1EDE            ; ln_Pred
0008        DC.B    8               ; ln_Type (8 = nt_Resource)
0009        DC.B    0               ; ln_Pri
000A        DC.L    FE478E          ; ln_Name -> 'misc.resource'
000E
000E        DC.B    6               ; lib_Flags
000F        DC.B    0               ; lib_Pad
0010        DC.W    C               ; lib_NegSize
0012        DC.W    32              ; lib_PosSize
0014        DC.W    20              ; lib_Version
0016        DC.W    A               ; lib_Revision
0018        DC.L    0               ; lib_IdString
001C        DC.L    0               ; lib_Sum
0020        DC.W    0               ; lib_OpenCnt
0022
0022 ;-------------------------   Zuordnungsliste (mr_AllocArray)
0022
0022        DC.L    0               ; UserName1
0026        DC.L    0               ; UserName2
002A        DC.L    0               ; UserName3
002E        DC.L    0               ; UserName4
0032
0032 ***************************************************************************
```

Misc.resource Routinen

```
4774 *************** Misc.Resource-Modul *********************************
4774
4774 FE4774  DC.W    4AFC                ; rt_Matchword
4776         DC.L    FE4774              ; rt_Matchtag
477A         DC.L    FE487C              ; rt_EndSkip
477E         DC.B    1                   ; rt_Flags
477F         DC.B    2Ø                  ; rt_Version
478Ø         DC.B    8                   ; rt_Type
4781         DC.B    46                  ; rt_Pri
4782         DC.L    FE478E              ; rt_Name
4786         DC.L    FE479C              ; rt_IdString
478A         DC.L    FE47B8              ; rt_Init
478E
478E FE478E  DC.B    'misc.resource',Ø
479C
479C FE479C  DC.B    'misc 32.1Ø (16 Jan 1986)',ØD,ØA,Ø,Ø
47B8
47B8 ;--------------------------------   Initialisierung
47B8
47B8 FE47B8  movem.l a2/a6,-(a7)         ; Register retten
47BC         movea.l 4,a6                ; a6 := SysBase
47C2         lea     FE486E,aØ           ; aØ -> Vektortabelle
47C8         lea     FE48Ø6,a1           ; a1 -> InitStruct-Tabelle
47CE         suba.l  a2,a2               ; a2 := Ø
47DØ         move.l  #32,dØ              ; dØ := 5Ø Bytes
47D6         jsr     -54(a6)             ; ---> MakeLibrary
47DA         tst.l   dØ                  ; Library ok?
47DC         bne.s   FE47F8              ; ja: ->
47DE         movem.l d7/a5-a6,-(a7)      ; Register retten
47E2         move.l  #22Ø2ØØØØ,d7        ; d7 := AlertKode
47E8         movea.l 4,a6                ; a6 := SysBase
47EC         jsr     -6C(a6)             ; ---> Alert
```

```
47F0          movem.l (a7)+,d7/a5-a6      ; Register wiederherstellen
47F4          bra     FE4800              ; ---> Ausgang
47F8
47F8 FE47F8   movea.l d0,a2               ; a2 := d0 -> misc.resource
47FA          movea.l a2,a1               ; a1 := a2
47FC          jsr     -1E6(a6)            ; ---> AddResource
4800 FE4800   movem.l (a7)+,a2/a6         ; Register wiederherstellen
4804          rts
4806
4806 ;--------------------------------   Tabelle für InitStruct
4806
4806 FE4806   DC.W    E000                ; InitByte
4808          DC.W    8                   ; ln_Type
480A          DC.B    8                   ; = nt_Resource
480B          DC.B    0
480C          DC.W    C000                ; InitLong
480E          DC.W    A                   ; ln_Name
4810          DC.L    FE478E              ; -> 'misc.resource'
4814          DC.W    E000                ; InitByte
4816          DC.W    E                   ; lib_Flags
4818          DC.B    6                   ; = Changed, Sumused
4819          DC.B    0
481A          DC.W    D000                ; InitWord
481C          DC.W    14                  ; lib_Version
481E          DC.W    20                  ; = 32
4820          DC.W    D000                ; InitWord
4822          DC.W    16                  ; lib_Revision
4824          DC.W    A                   ; = 10
4926          DC.W    0
4828          DC.W    0
482A
482A ;--------------------------------   GetMiscResource
482A
482A FE482A   movea.l 4,a0                ; a0 := SysBase
4830          lsl.l   #2,d0               ; d0 := MRType * 4
4832          move.l  d0,d1               ; d1 := d0 -> UserName-Offset
4834          movea.l 4,a0                ; a0 := SysBase (warum nochmal?)
4838          move.w  #4000,DFF09A        ; Interrupts sperren
4840          addq.b  #1,126(a0)          ; IDNestCnt erhöhen
4844          move.l  22(a6,d1.w),d0      ; d0 -> UserName
```

```
4848          bne.s   FE484E              ; Resource belegt: ->
484A          move.l  a1,22(a6,d1.w)      ; UserName eintragen
484E FE484E   movea.l 4,aØ                ; aØ := SysBase
4852          subq.b  #1,126(aØ)          ; IDNestCnt erniedrigen
4856          bge.s   FE486Ø              ; Interrupts noch gesperrt: ->
4858          move.w  #CØØØ,DFFØ9A        ; Interrupts freigeben
486Ø FE486Ø   rts
4862
4862 ;--------------------------------   FreeMiscResource
4862
4862 FE4862   lsl.l   #2,dØ               ; dØ := UnitNum * 4 = UserName-Offset
4864          move.l  #Ø,22(a6,dØ.w)      ; UserName löschen
486C          rts
486E
486E ;--------------------------------   Vektortabelle für Sprungliste
486E
486E FE486E   DC.L    FE482A              ; GetMiscResource
4872          DC.L    FE4862              ; FreeMiscResource
4876          DC.L    -1                  ; Endemarke
487A
487A          DC.W    Ø
487A
487A **************************************************************************
```

Register

Allgemeine Routinen

AbortIO 48
BeginIO 46
Close 50
Delayed Expunge 50
Device einrichten 49
Device entfernen 49
Flush 51
IORequest beantworten 48
IORequest senden 47
Reset 51
Signal freigeben 52
Signal zuweisen 52
Start 51
Stop 51

Console.device

AskDefaultKeyMap 144
AskKeyMap 144
Byte aus Puffer 126
Byte in Puffer 125
CDInputHandler 139
Clear 127
Close 148
CSI-Sequenzen Tabellen 176,191,194
Ctrl-Kodes Tabelle 171
Datenbereich 116
Display ab Kursor löschen 127

Console.device (Forts.)

Esc-Sequenzen Tabelle 174
Expunge 138
Header 46
Initialisierung 134
InputEvent-Report 141
Kommandotabelle 137
Kursor ausschalten 130
Kursor einschalten 129
Kursor löschen 130
Kursor nach oben 131
Kursor nach unten 132
Kursor setzen 131
Kursor umschalten 129
Kursor umsetzen 131
Kursor vorwärts 132
Kursor zurück 133
LockDisplay 123
Open 145
RawKeyConvert 148
Read 154
ReadRequest bearbeiten 141
Rechteck löschen 128
Scroll Display 133
SetDefaultKeyMap 144
SetKeyMap 144
SGR-Tabelle 185
Tabulator-Routinen 155
UnlockDisplay 124

Console.device (Forts.)

Vektortabelle 137
Window-Routinen 157
Wort aus Puffer 126
Wort in Puffer 125
Write 163
Zeile ab Kursor löschen 127

CIA.Resource

AbleICR 315
AddICRVector 314
ConUnit 119
Datenbereich A 305
Datenbereich B 308
Header 311
Initialisierung 312
InterruptServer 313
RemICRVector 315
SetICR 315
Vektortabelle 311

Disk.resource

AllocUnit 330
Datenbereich 322
FreeUnit 330
GetUnit 329
GetUnitID 331
GiveUnit 330
Header 325
Initialisierung 325
InterruptServer 331
Vektortabelle 332

Gameport.device

AskCType 79

Gameport.device (Forts.)

AskTrigger 82
Clear 76
Datenbereich 68
Expunge 75
Header 45
Initialisierung 72
InputHandler 83
Kommandotabelle 75
Open 76
ReadEvent 77
SetCType 79
SetTrigger 82
Vektortabelle 75

Input.device

AddHandler 104
Clear 104
Datenbereich 90
Expunge 103
Header 45
Initialisierung 98
InputHandler rufen 115
InputTask 107
Kommandotabelle 102
Open 103
RemHandler 105
Reset 103
SetMPort 106
SetMTrig 106
SetMType 106
SetPeriod 105
SetThresh 105
Start 104
Vektortabelle 102
WriteEvent 105

Keyboard.device

AddResetHandler 64
Clear 61
Datenbereich 54
Expunge 61
Handshake 58
Initialisierung 57
InputHandler 65
Kommandotabelle 60
Open 61
ReadEvent 62
ReadMatrix 63
RemResetHandler 64
Reset 60
ResetHandlerDone 65
Vektortabelle 60

Keymap.resource

Header 44
Initialisierung 197
Keymap D 205
Keymap USA 197

Misc.resource

Datenbereich 333
FreeMiscResource 336
GetMiscResource 335
Header 334
Initialisierung 334
Vektortabelle 336

Potgo.resource

AllocPotBits 319
Datenbereich 317
FreePotBits 320
Header 318
Initialisierung 318
Vektortabelle 319
WritePotgo 321

Trackdisk.device

AddChangeInt 249
BeginIO 259
Bytes lesen 272
Bytes schreiben 273
ChangeNum 248
ChangeState 248
Clear 283
Close 258
Daten dekodieren 287
Daten kodieren 284
Datenbereich 237
Expunge 257
Format 261
GetDriveType 303
GetNumTracks 304
Header 245
Initialisierung 250
Kommandotabelle 268
Kommandoverteiler 260
Kopf auf Spur 0 268
Kopf auf Spur d0 269
Kopfschritt 271
Laufwerk anfordern 290
Laufwerk freigeben 291
Laufwerkprüfung 246
Motor 282
Motor schalten 270
Open 258
Parametertabellen 267
ProtStatus 282
Prüfsumme berechnen 290
RawRead 301

Trackdisk.device (Forts.)

RawWrite 302
Read 276
RemChangeInt 249
Remove 248
ResetHandler 257
Seek 282
SektorHeader suchen 293
Spur lesen 281
Spur normalisieren 295
Spur prüfen 295
Spur schreiben 280
TDUnit Datenbereich 240
Trackdisk-Task 291
Unit erzeugen 253
Update 283
Vektortabelle 267
Write 276

Timer.device

AbortIO 229
AddRequest 227
AddTime 235
BeginIO 223
Close 225
CmpTime 235
Datenbereich 213
Expunge 223
GetSysTime 225
Header 217
Initialisierung 218
InterruptServer 230
Kommandoverteiler 223
Open 224
SetSysTime 226
SubTime 234
Vektortabelle 217

COMMODORE

AMIGA KNOWHOW

1

DR. RUPRECHT

KOMMENTIERTES ROM-LISTING

Ein **Profi-Sachbuch** (genauer gesagt eine Sachbuchreihe) mit der detaillierten Dokumentation des **AMIGA-Betriebssystems.**

Der **erste Band** des **AMIGA-KNOWHOW** umfaßt ca. 300 Seiten und dokumentiert den Kickstart-Bereich mit EXEC, BOOT-ROM und DOS-BOOT. In diesem Bereich liegen z.B. auch die Multi-tasking Routinen des AMIGA.

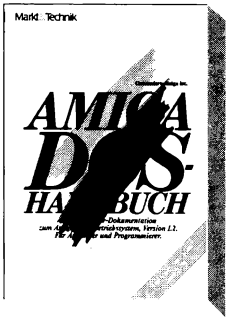
Der **Autor** dieser Bücher, **Dr. Ruprecht**, ist als einer der intimsten Kenner von Commodore-Betriebssystemen anerkannt und hat sich bereits in den vergangenen Jahren durch mehrere Dokumentationen einen Namen gemacht.

Dr. Ruprecht war außerdem an der Entwicklung des legendären SM-KIT für CBM 3032, 8032, 8096, C64 und C128 sowie an der Entwicklung des Betreibssystems LOS 96 für den CBM 8096 (dem Orginal Commodore Betriebssystem) maßgeblich beteiligt.

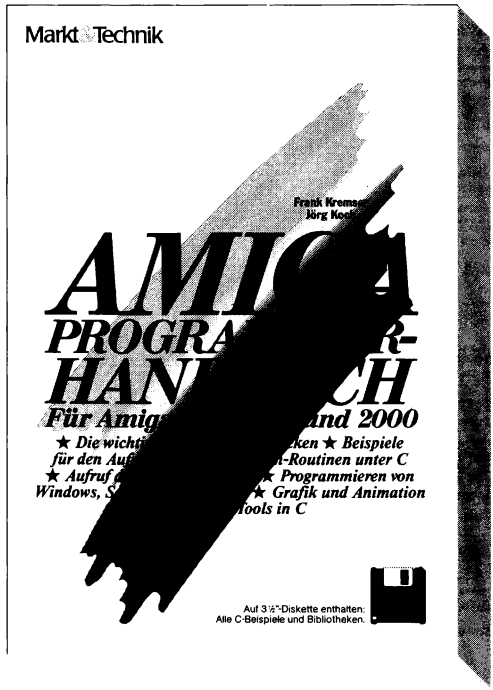
Der Band AMIGA-KNOWHOW kostet DM 69,-
(unverb. empfohlener Verkaufspreis)

Vertrieb: BIOSYSTEMS SRI GmbH,
Hansjakobstr.122 · 8 München 82
und Mediscript-Verlag München.

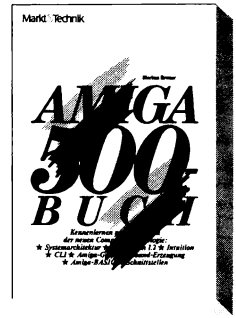
Bücher zum Amiga



Bantam Books
Das Amiga-DOS-Handbuch für Amiga 500, 1000 und 2000
1987, ca. 300 Seiten
Die Pflichtlektüre für jeden Commodore-Amiga-Anwender und Programmierer: eine Entwickler-Dokumentation zum Amiga-DOS-Betriebssystem, Version 1.2. Programmierung, interne Datenstruktur und Diskettenhandling. Mit diesem Buch lernen Sie das mächtige Amiga DOS schnell und sicher zu beherrschen. Alle Möglichkeiten des Systems, bis hin zum »Multi-Tasking« werden ausführlich und anschaulich beschrieben.
Best.-Nr. 90465
ISBN 3-89090-465-3
ca. DM 59,–



Kremser/Koch
Amiga Programmierhandbuch
1987, ca. 300 Seiten, inkl. Diskette
Eine tolle Einführung in die »Interna« des Amiga: Die wichtigsten Systembibliotheken, die das Betriebssystem zur Verfügung stellt, werden anhand vieler Beispiele erklärt. Aus dem Inhalt: Aufruf der Betriebssystem-Routinen unter C, Aufruf der DOS-Funktionen, Programmieren von Windows, Screens und Gadgets, Grafik und Animation, Tips und Tools in C.
Best.-Nr. 90491
ISBN 3-89090-491-2
DM 69,–



M. Breuer
Das Amiga-500-Handbuch
1987, ca. 450 Seiten
Eine ausführliche Einführung in die Bedienung des Amiga 500. Kennenlernen und Anwenden der neuen Computer-Technologie: Systemarchitektur, Workbench 1.2, Intuition, CLI, Amiga-Grafik, Sound-Erzeugung, Amiga-BASIC und Schnittstellen. Neben dem Handbuchteil mit vielen Bildschirmfotos und Übersichtstabellen, die Ihnen beim täglichen Einsatz helfen, schnell und reibungslos zu arbeiten, enthält das Buch eine ausführliche Beschreibung des Amiga 500 und seines Zubehörs.
Best.-Nr. 90522
ISBN 3-89090-522-6
DM 49,–

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

707328

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0

COMMODORE

AMIGA

500/1000/2000

KNOW HOW

DR. RUPRECHT

KOMMENTIERTES ROM-LISTING

Der Autor Dr. Ruprecht gilt als einer der anerkanntesten Kenner der Commodore Betriebssysteme. Zu fast allen Commodore-Geräten hat er umfangreiche Betriebssystem-Dokumentationen erarbeitet und veröffentlicht.

Beim legendären CBM 8096 war er selbst maßgeblich an der Entwicklung des Betriebssystems LOS 96 beteiligt.

Byte für Byte hat der Autor die jeweiligen Stellen des Betriebssystems aufgelistet und mit Kommentaren versehen und damit das, was im Amiga-Betriebssystem abläuft, transparent und nachvollziehbar gemacht.

Der vorliegende 2. Band ist eine Dokumentation aller Resources und der wesentlichen Devices des Amiga-Betriebssystems. Dazu gehören die Beschreibungen der Devices: **DISK, KEYBORD, GAMEPORT** (Maus + Joystick), **INPUT, CONSOLE, TRACK-DISK und TIMER.**

Ein unentbehrliches Sachbuch und Nachschlagewerk für jeden, der sich professionell mit dem Amiga beschäftigt.

Bisher erschienen vom Autor:

Rom Listing CBM 3000	bei SM Software
Rom Listing CBM 8000	bei SM Software
Rom Listing CBM 8050	bei SM Software
Rom Listing CBM 8096	bei SM Software
Rom Listing +4 / C 16	bei Commodore
Rom Listing C 128	bei Markt & Technik
Rom Listing AMIGA, Bd. 1	bei Mediscript

Eine Gemeinschaftsproduktion von:

EDOTRONIK®

GIV Gesellschaft für Informations-verarbeitung mbH

BIOSYSTEMS SRI GmbH

Erschienen bei: Mediscript-Verlag · München

ISBN 3-88320-169-3

This was brought to you

from the archives of

http://retro-commodore.eu